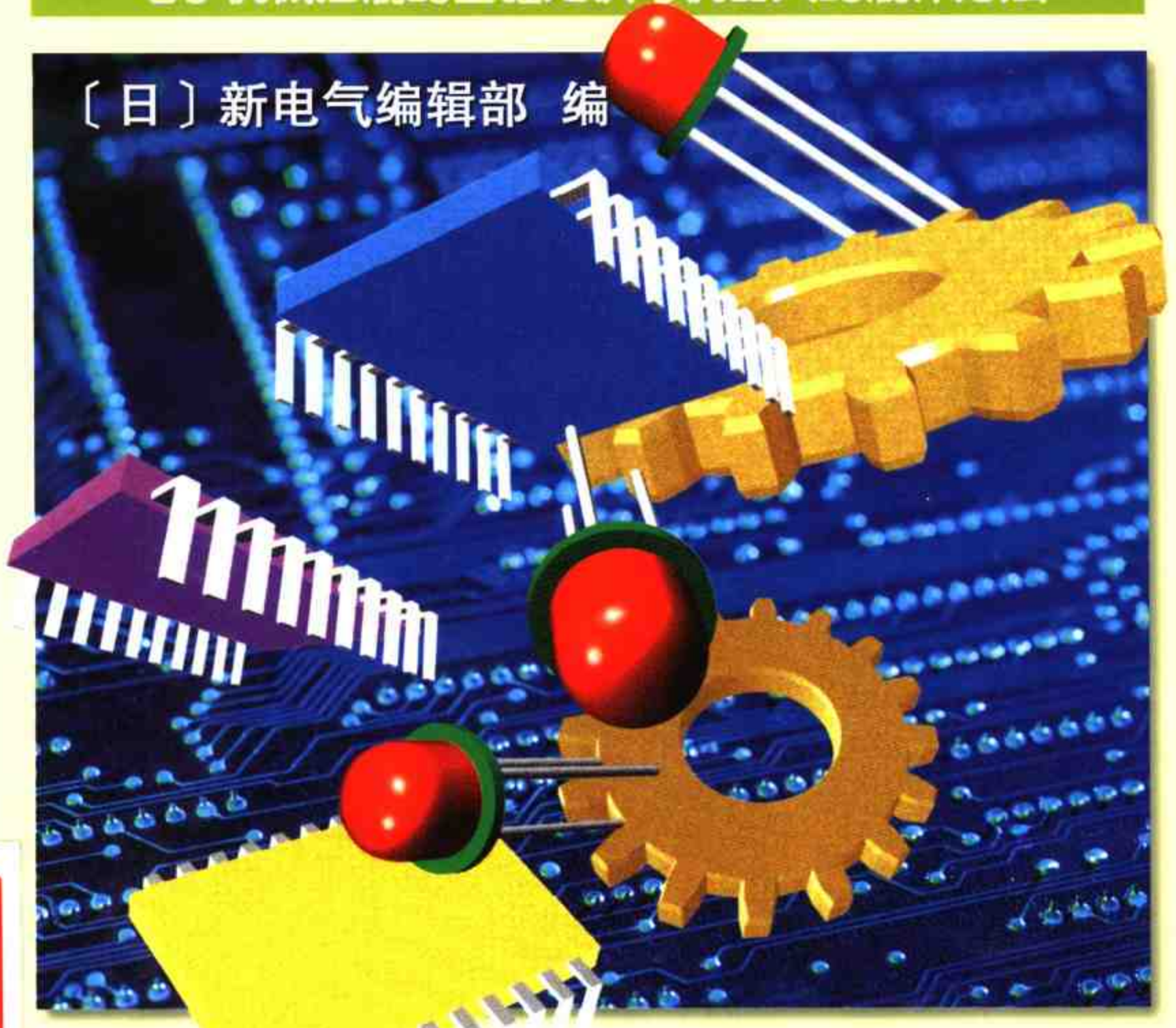
OHM 电子爱好者读物

电子控制入门

电子机械控制的基础知识与机器人的制作方法

〔日〕新电气编辑部 编



科学出版社
www.sciencep.com

(TN-0443.0101)
责任编辑　崔炳哲
责任制作　魏　谨
责任印制　刘士平
封面设计　李　力

◇◇◇◇◇◇◇ 内　容　简　介 ◇◇◇◇◇◇◇

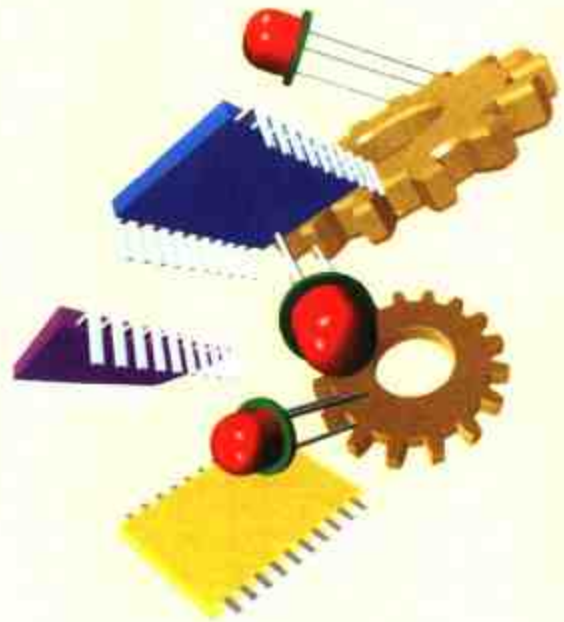
“OHM电子爱好者读物”系列共有6本，涉及电子电路、电子机械、电子控制、数字电路、机电一体化、电子技术等。本系列以初学者为对象，以实用技术为重点，利用丰富的插图，甚至建立模拟教室以对话的形式、通俗易懂地介绍相关内容。阅读本系列的读者最好边学习边实践，以使所学知识变成自己有用的技能。

本书是“OHM电子爱好者读物”之一。书中首先介绍传感器与执行机构等控制所需的基本知识与技术，然后以信号灯顺序控制及相扑机器人制作为例，讲述了计算机控制基础知识，内容包括各种控制用微机、接口、控制程序编程语言等知识及其应用。

本书可供计算机控制相关专业的大专院校学生学习使用，亦可供从事自动控制领域的技术人员参考阅读。

OHM 电子爱好者读物

电子电路入门
数字电路入门
电子机械入门
电子电路与电子技术入门
电子控制入门
机电一体化入门



ISBN 7-03-011087-0
9 787030 110879 >

ISBN 7-03-011087-0
定　价: 20.00 元

OHM电子爱好者读物

电子控制入门

电子机械控制的基础知识与机器人的制作方法

〔日〕新电气编辑部　编
崔东印　译

科学出版社
北京

图字：01-2003-0411 号

Original Japanese language edition
Hajimete Manabu Denshi Seigyo Nyumon Hayawakari
Edited by Shindenki Henshubu
Copyright © 1998 by Ohmsha, Ltd.
Published by Ohmsha, Ltd.
This Chinese version published by Science Press, Beijing
Under license from Ohmsha, Ltd.
Copyright © 2003
All rights reserved

初めて学ぶ
電子制御入門早わかり
オーム社 1998

图书在版编目(CIP)数据

电子控制入门/(日)新电气编辑部编；崔东印译. - 北京：科学出版社，2003
(OHM 电子爱好者读物)

ISBN 7-03-011087-0

Ⅰ.电… Ⅱ.①新…②崔… Ⅲ.电子控制-基本知识 Ⅳ.TN.1

中国版本图书馆 CIP 数据核字(2002)第 106822 号

责任编辑 崔炳哲 **责任制作** 魏 谨
责任印制 刘士平 **封面设计** 李 力

科学出版社 出版
北京东黄城根北街 16 号 邮政编码：100717
http://www.sciencep.com

源海印刷有限责任公司 印刷

北京东方科龙图文有限公司 制作
http://www.okbook.com.cn

科学出版社发行 各地新华书店经销

2003 年 7 月第 一 版 开本：B5(720×1000)
2003 年 7 月第 次印刷 印张：12
印数：1—5 000 字数：197 000

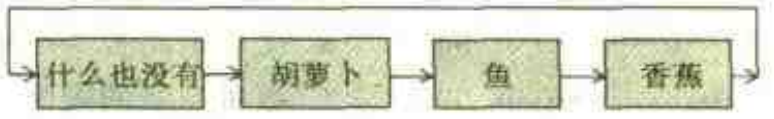
定 价：20.00 元

(如有印装质量问题，我社负责调换(新欣))

卷首插图 皮耶罗魔术师的“机关”



卷首插图所示的魔术节目中，中心部分是台子上的东西按下述顺序变化的魔术用“机关”。



利用同样的“机关”，1900 年法国的鲁诺发表了名为“可爱的魔术师”的木偶。在日本，1796 年细川半藏编写的“机巧图汇”中已有名为“品玉人形”的魔术木偶的制作记录。总之，过去的“机关”木偶，都是熟练的工匠靠着精工细作的技艺制作出来的。“皮耶罗魔术师”的“机关”，不仅与微机控制很相似，而且也考虑到制作上尽可能地简单（台上东西变换的机关如图 1 所示）。



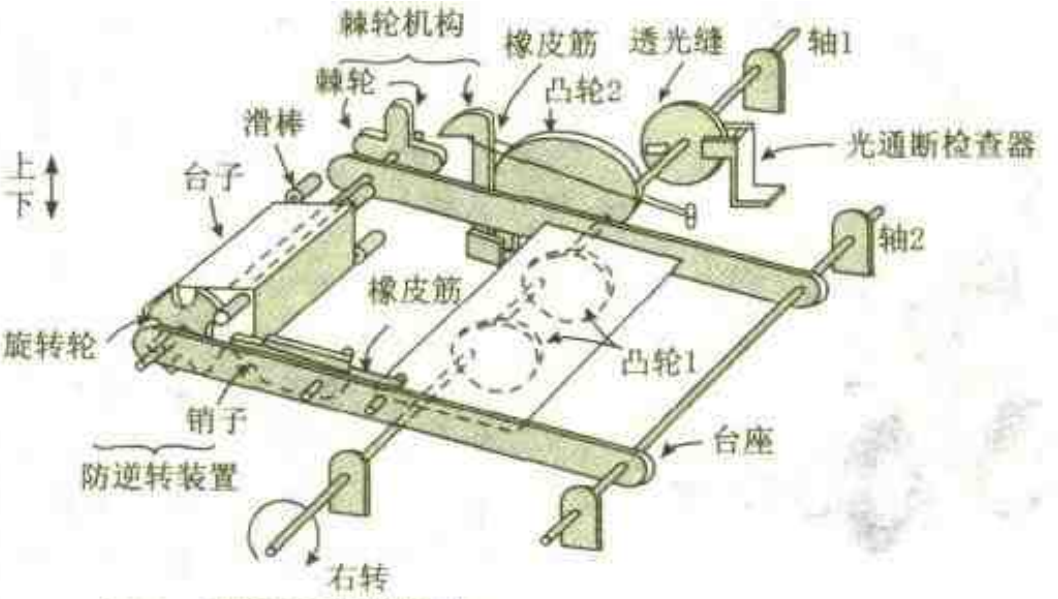


图 1　调换东西的机构

电机使轴 1 旋转，由于凸轮 1 的作用，台座以轴 2 为支点上下转动。而且当台座要向下时，由于凸轮 2 的作用，棘爪离开棘轮，而台座要向上时，棘爪与棘轮啮合，使台子向右回转。即台子向上运动的同时，由于棘轮机构的作用，台子会如图 1 所示向右回转。回转轮与销子组成防逆转装置，以防止台座向下时台子的偶然回转。轴 1 的回转位置（台子的高度），使用光通断检查器进行检测（台子回转嵌入地板的顺序如图 2 所示）。

① 从舞台上看到的台子的 A 面；

② 与①相同的状态下，从侧面看到的台子的样子；

③ 台子下降；

④ 台子右转的同时上升；

⑤ 只要台子转过 45°以上，台子四角的滑棒就会滑入地板；

⑥ 台子被引导到确定位置，B 面被正确嵌入地板。

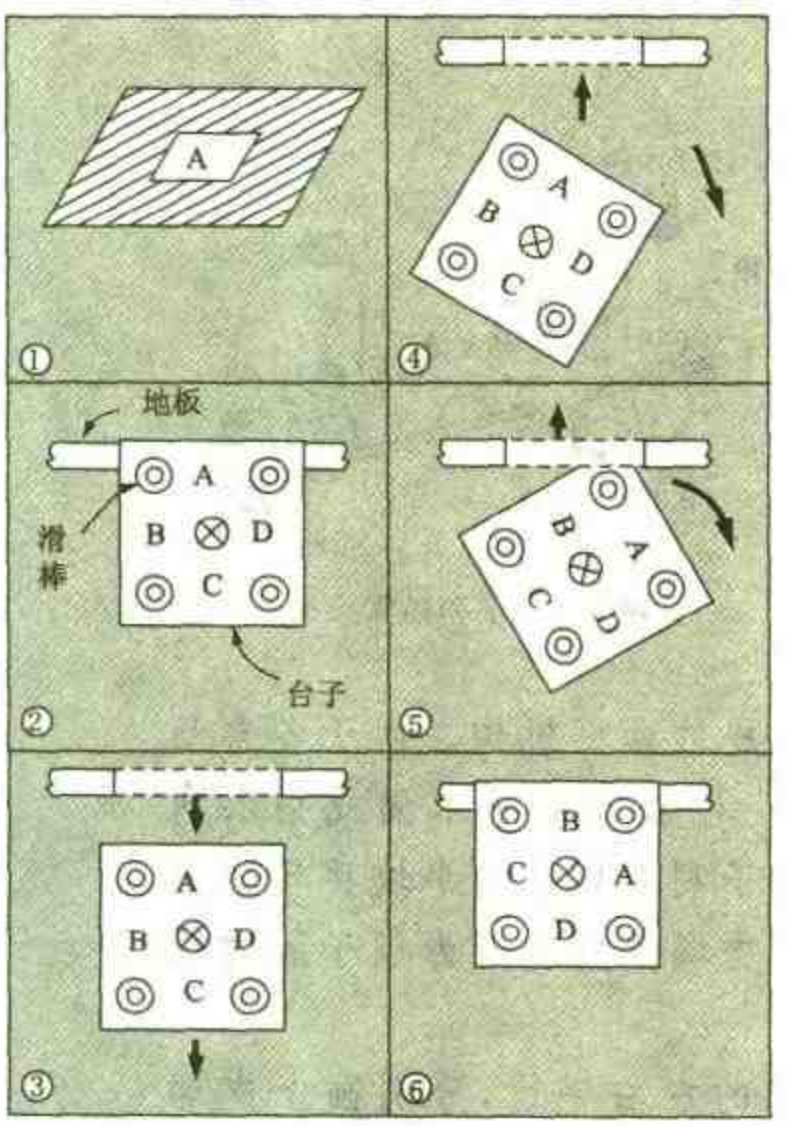


图 2　台子的回转机构

以后，重复同样的动作，台子即可按C、D、A的顺序变化登场（皮耶罗举、放箱子的机关如图3所示）。

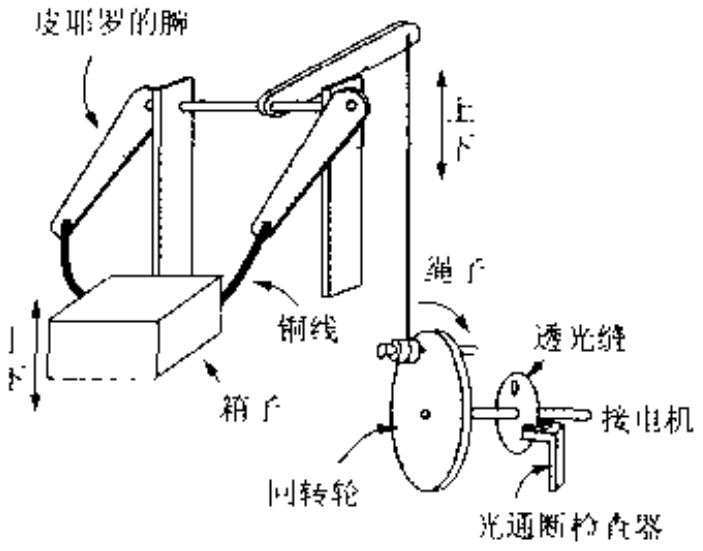


图3　举箱子机构

这是一个简单的曲柄机构。电机使轴回转，使得皮耶罗的脸上下。脸上下状态的位置，用光通断检查器进行检测（兔子和猫露头机关如图4所示）。

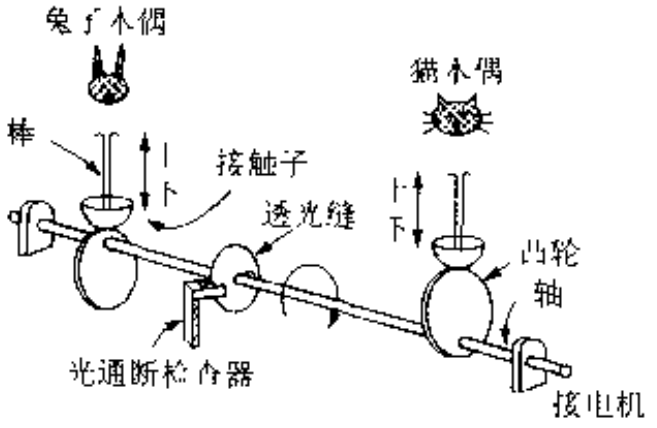


图4　兔子和猫露头机关

这是两只凸轮构成的曲柄机构。由于各自凸轮的作用，会使得其上端放有兔子木偶或猫木偶的棒上下运动。接触子利用切开的半块乒乓球做成，以减小摩擦阻力（皮耶罗的脸变为猴子脸的机关如图5所示）。

在硬泡沫塑料球的左右半边，分别画上皮耶罗和猴子的脸，用电机使其旋转即可。

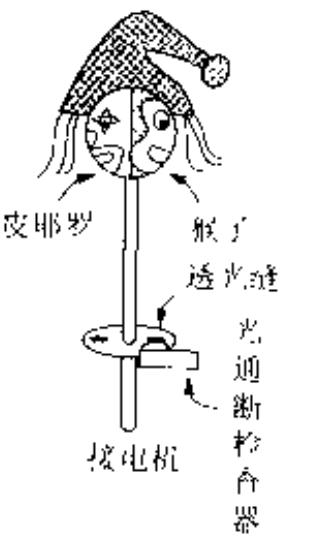


图5　皮耶罗变脸机构

本书著作权和专有出版权受到《中华人民共和国著作权法》的保护。凡对本书的一部分或全部进行转载，或用复印机进行复制或在其他场合引用，以及录入电子设备等行为，均属侵害著作权，构成违法。

本书如需复制、引用、转载、改编时，必须得到版权所有者的许可。

如有任何疑问请与以下部门联系。联系时请尽量使用信函或传真形式。

科学出版社总编部

电话：010－64012994　传真：010－64019810

读者服务部：010－64017892　010－64000246

邮政编码：100717　　地址：北京市东黄城根北街 16 号

http://www.sciencep.com

北京东方科龙图文有限公司

编辑部电话：010－82857401

促销部电话：010－82843276 转 219，220　传真：010－82842304

邮政编码：100029　　地址：北京市朝阳区华严北里 11 号楼 3 层

http://www.okbook.com.cn

目 录



第 1 章 电子控制的基础知识 1

1 传感器与执行机构 …… 2
2 光传感器与温度传感器 …… 5
3 磁传感器与压力传感器 …… 8
4 电磁螺线管与步进电机 …… 11
5 伺服电机及其应用 …… 14
6 接口电路 …… 17
7 A-D 转换与 D-A 转换 …… 20

第 2 章 C 语言控制入门 24

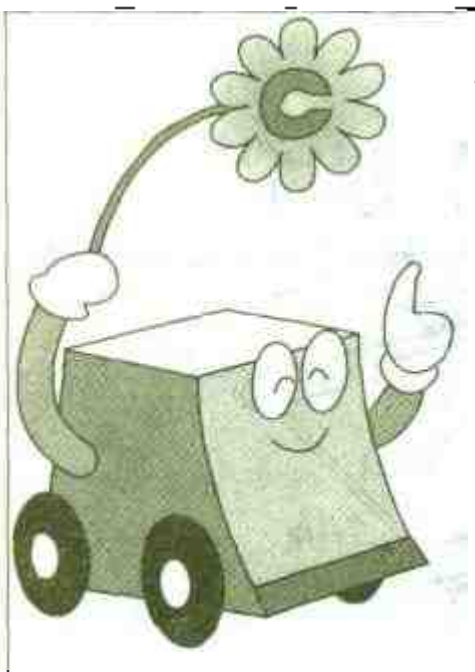
1 LED 的闪烁控制 …… 25
2 LED 闪烁速度的调整与音阶 …… 54
3 LED 闪烁的缓慢停止 …… 79

第 3 章 计算机控制基础 93

1 微机控制的种类 …… 94
2 微机控制的实例 …… 99
3 微机电路板 …… 104
4 微机控制的顺序 …… 109

第 4 章 机器人的制作方法 116

1 机器人相扑比赛大会概况 …… 117
2 自立型相扑机器人“太郎” …… 122
3 微机板的使用方法 …… 126
4 机械部分的制作方法 …… 130
5 电子电路的制作方法 …… 141
6 执行文件的制作方法 …… 149
7 ROM 写入器的使用方法 …… 156
8 遥控调试程序的使用方法 …… 161
9 程序调试实践 …… 167
10 “太郎”的程序 …… 174



本书是按照工业高等学校* 电气、电子以及电子机械专业所开设的“电子机械”课程的教学大纲编写的简明计算机控制入门书。

书中首先介绍作为电子机械控制基础的传感器及执行机构的基础，然后介绍 C 语言控制的基础知识。

本书的后半部分，作为计算机控制的基础知识及其应用，介绍相扑机器人的制作方法。

新电气编辑部

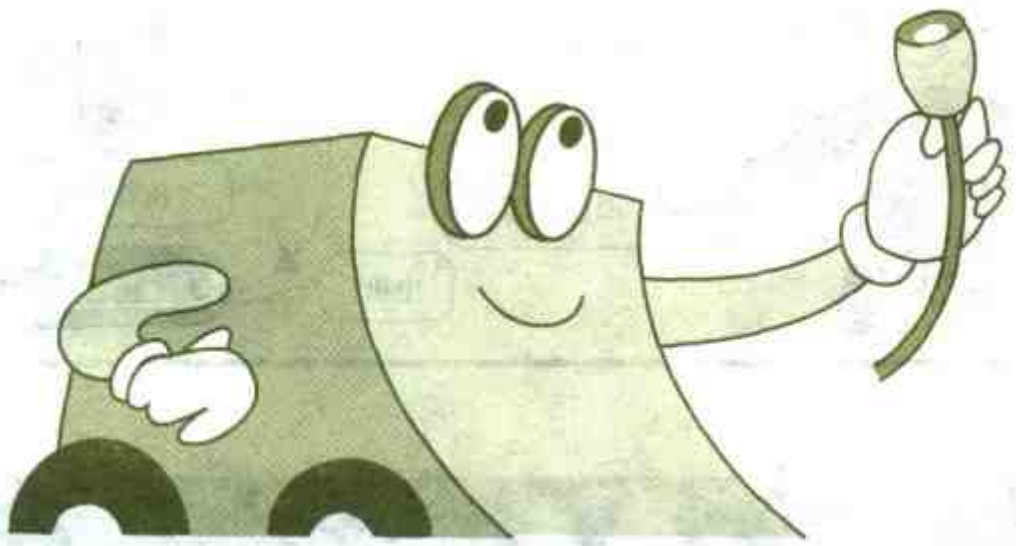
本书各章执笔者分别为

第 1 章	全国工业高等学校校长协会，顾问	岩 本 洋
第 2 章	新潟县立柏崎工业高等学校，教师	小野塚纯夫
第 3 章	东京都立藏前工业高等学校，教师	堀 桂太郎
第 4 章	东京都立藏前工业高等学校，教师	堀 桂太郎
	新潟县立新潟工业高等学校，教师	村田 千夫

* 日本的高等工业学校相当于我国的中专或技校。

第 1 章

电子控制的基础知识



本章学习内容

将光、压力、气体等各种物理量以及化学量变为电量的传感器技术很早就有,随着半导体技术的进步,其研究开发获得了长足的发展。

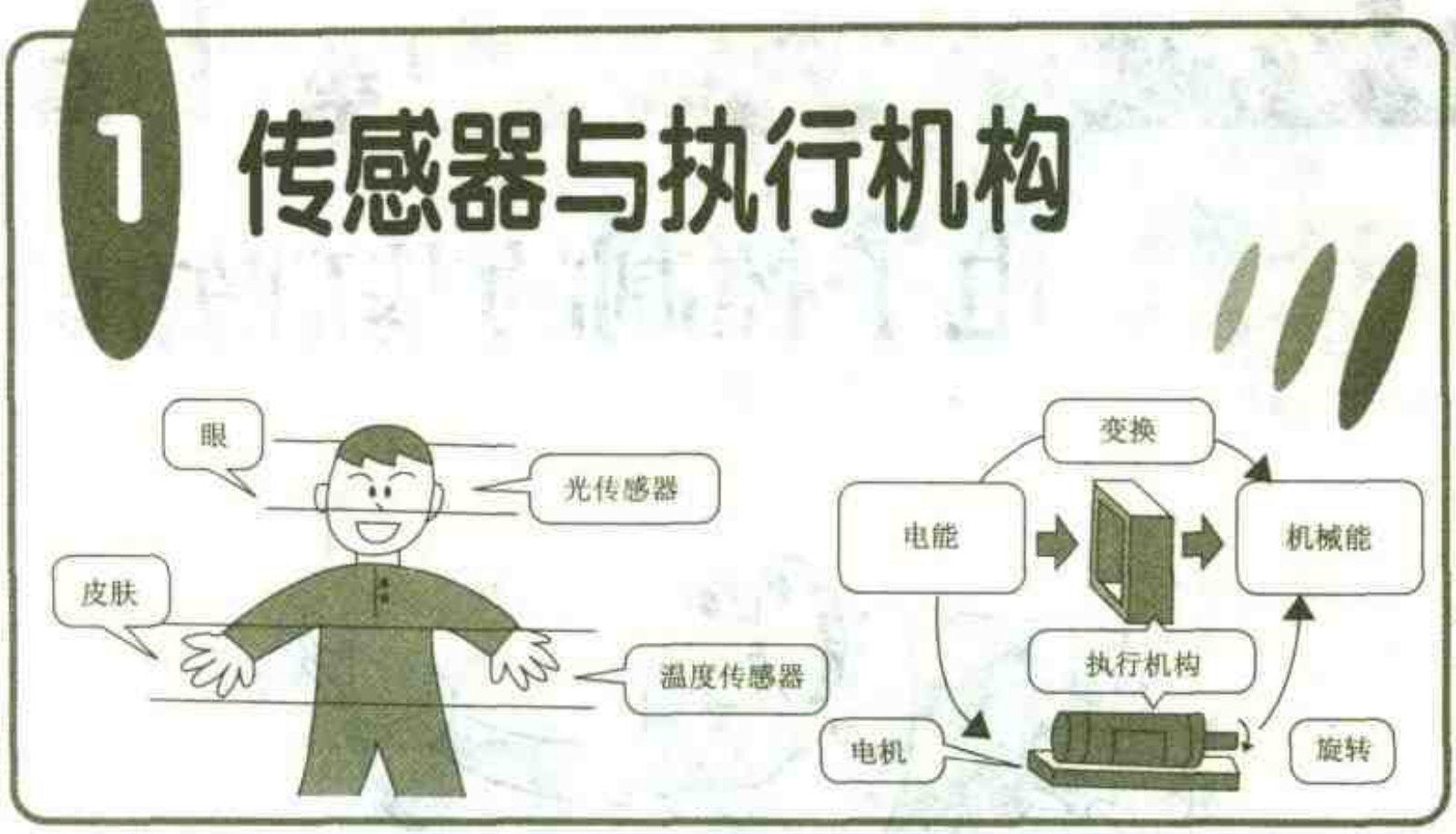
另外,步进电机以及电磁继电器等执行机构也在进步,各种计算机控制中,种种控制方法得到了利用。

而最初的计算机被衣阿华州大学(美国)的阿塔那索夫称为ABC机,它作为现代计算机的雏形被制造出来以后,世界上相继作出了ENIAC、EDSAC等计算机,奠定了现代计算机的基础。之后还开发了用于机器人以及家电等产品的称为微处理器的小型计算机。

为了用计算机驱动执行机构工作,必须将计算机的输出信号变为驱动执行机构工作所必需的信号。因此要在计算机与执行机构之间设置接口电路,以仔细调整信号的形式及大小等。

本章介绍传感器与执行机构的种种知识,以及接口电路的基本原理。电子控制的基础知识,就是通过这些技术的学习逐渐积累起来的。

1 传感器与执行机构



Ⅰ　传感器

“传感器”一词现在经常使用。在日本工业标准(JIS)中，传感器被定义为“为实现感觉功能的检测器件”。

这里所谓感觉功能，指的是人的视觉、听觉、味觉、嗅觉、触觉等五种感觉。

为了使机器人之类的计算机控制装置工作，相当于人类感觉功能的检测器件是必需的。这种器件称为**传感器**。

图 1 所示为用传送带移动瓶子，并自动检测瓶子的个数及瓶子中液量的示意图。

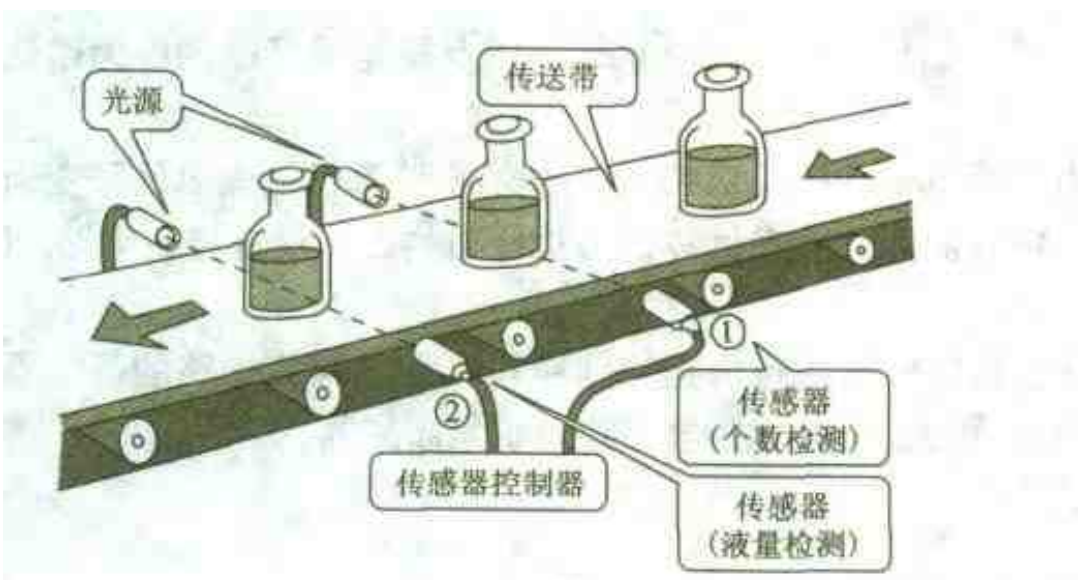


图 1　传感器

传感器①检测个数，传感器②检测液量。传感器就是这样被利用的。

传感器有各种各样，典型的传感器有：

光传感器 用于检测眼睛能看到的光(可见光)以及看不见的光(紫外线、红外线)。

温度传感器 用于检测物体发出的热以及由热产生的辐射线等。

压力传感器 给物体施加压力即会发生变形。压力传感器用来检测其变形的程度。

气体传感器 用于检测气体的泄漏量及气体的种类。

磁传感器 用于检测是否有磁。

Ⅱ 执行机构

把电能变为机械能而做功的机器称为**执行机构**。例如，将电能变为回转运动的电动机是驱动器，将气压、油压等变为直线运动的气缸、油缸也是执行机构。

图 2(a)为人手要抓球的示意图。为抓球，要将指头移近球，而且必须适当地运动肌肉。

图 2(b)为机器人的手要抓球的示意图。为抓球，电动机和气缸必须要有适当的动作。

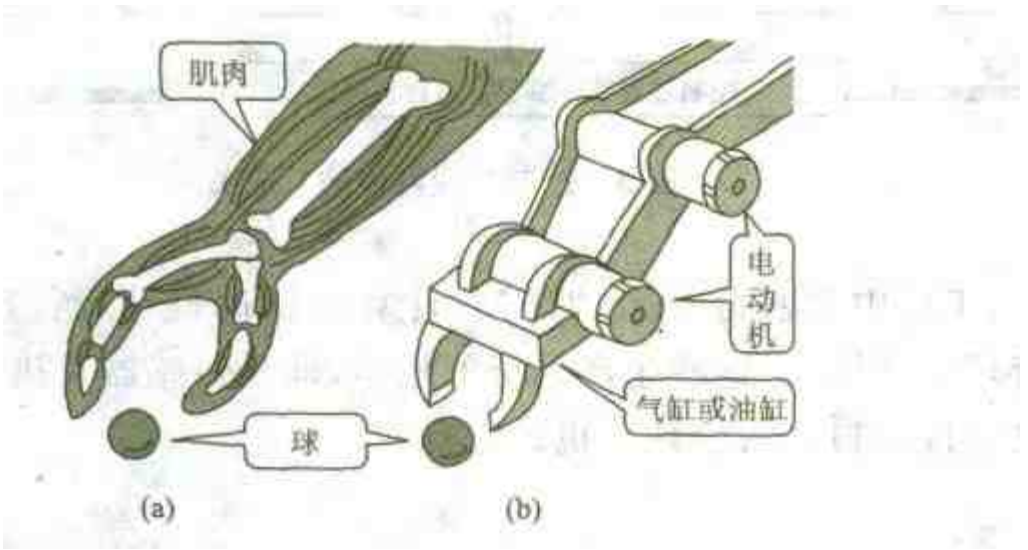


图 2 执行机构

根据驱动能源的不同，执行机构被分为电气型、气压型、油压型等。

- 电气型
 - 电磁铁 —— 电磁螺线管
 - 电动机 —— 伺服电机，步进电机
- 气压型 —— 气压马达，气缸
- 油压型 —— 油压马达，油缸

Ⅲ　控制装置与人体部位的对应关系

为了看电视节目，要拨开关、调频道、调音量。这种为满足用户的需求而进行必要的操作，称之为**控制**。

图 3 示出的是控制装置与人体部位的对应关系。

人是以耳闻目睹的信息为基础，用头脑思考、判断，再用手足进行动作。这时信息作为一种信号，通过神经系统进行交换。控制系统中，与此神经系统相对应的，即是接口。在计算机与控制对象间连接电子电路，以调整信号的大小及形状的电子电路称为**接口**。

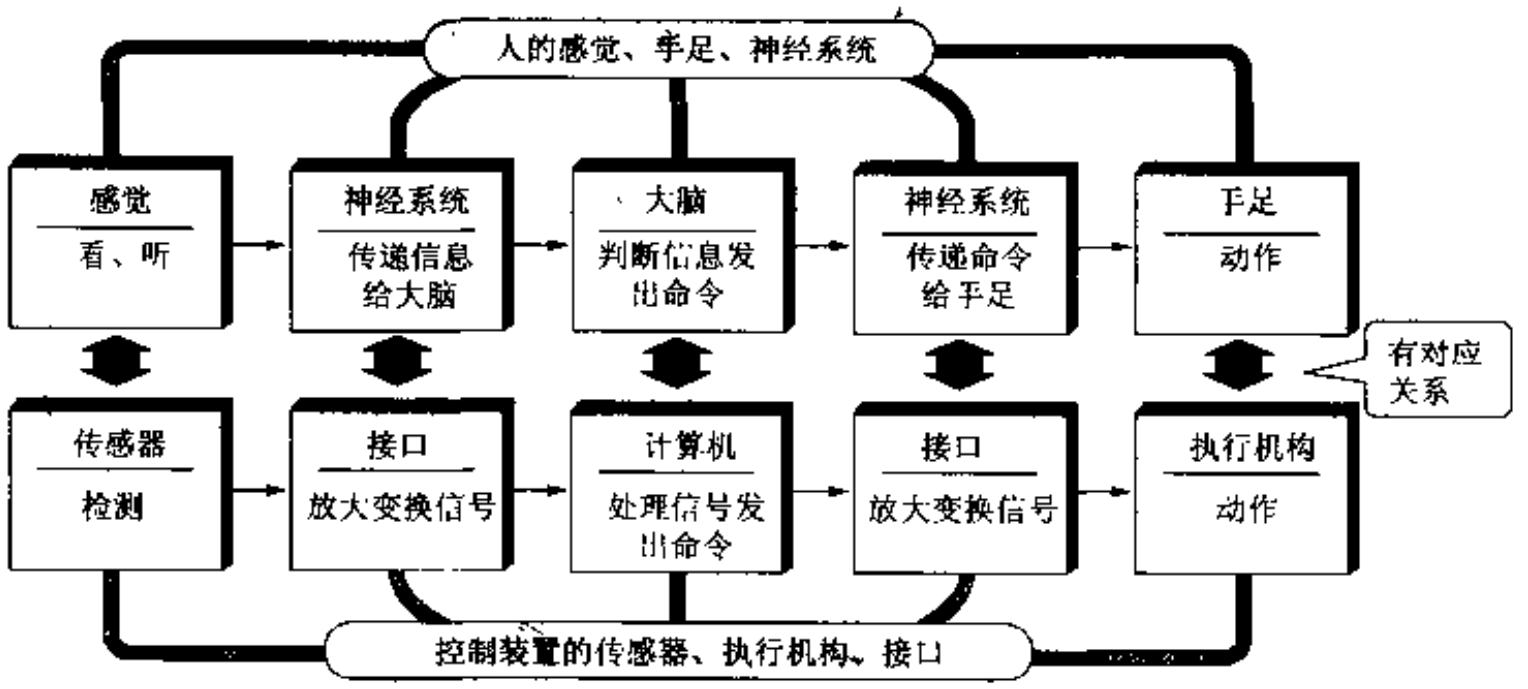


图 3　控制装置与人的对应关系

现在，工厂中很盛行 CAD/CAM（计算机辅助设计/制造）及 CIM（含制造与生产管理的系统）。这些生产系统的根本，即是传感器及执行机构，来自传感器的信息通过接口输入给计算机。

思考题

●将适当的用语填入下面句子的（　）中，使句子正确。

（1）为实现感觉功能的检测器件称为（①）。

（2）控制装置中，气缸及电动机等将电能变换为机械能的作功机器称为（②），它们被分类为（③）型、（④）型、（⑤）型。

（3）计算机、控制对象与传感器中处理的信号是不同的，因而，必须要有（⑥）。

2 光传感器与温度传感器



I 光传感器

半导体器件中，有光电二极管、光电三极管、发光二极管(LED)、CdS 等等。使用这些元器件或其组合，可以制成检测光的**光传感器**。

图 1 是光电三极管及其特性。给以一定的照度(例如 100 lx)，随着集电极电压的逐渐上升，如图中曲线所示，光电流随之增加。如图所示，照度不同，V_C-I_C 特性不同。具有这种特性的光电三极管，常用作**光的接收器件**。

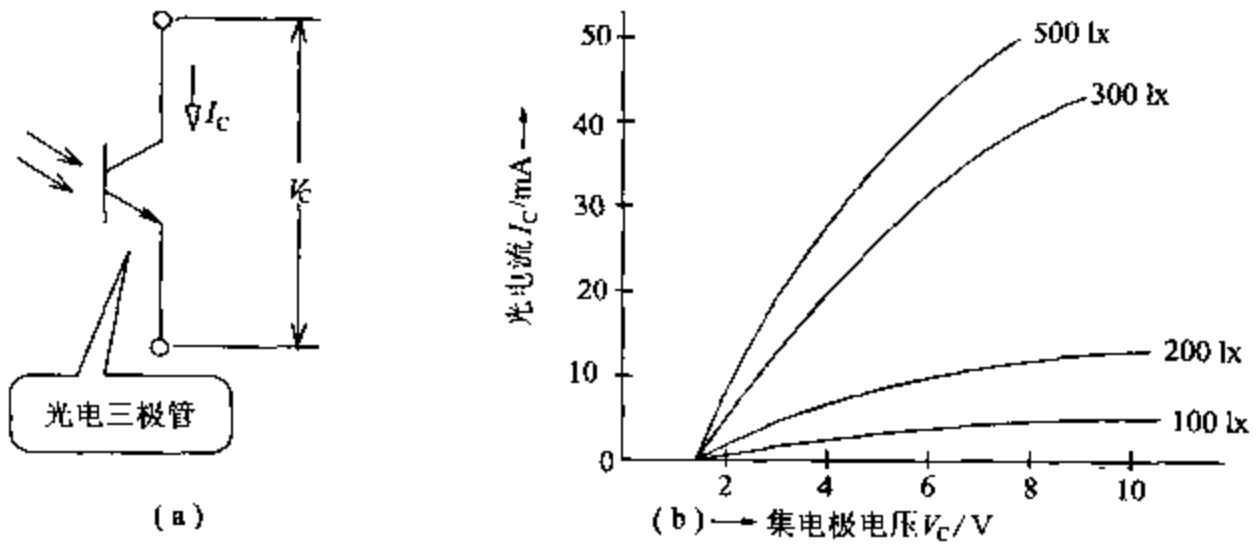


图 1　光电三极管及其特性曲线

Ⅱ　光电开关

光电开关是检测物体是否接近的装置，它由发光元件与接收元件组合而成。光电开关分为**透射型**与**反射型**两类。

图 2 为透射型光电开关。泛光灯有 LED，用来发光。发出的光由作为接收器的光电三极管接收。一旦物体靠近遮断了光，三极管即可检测出这种状态。

图 3 为反射型光电开关。物体靠近时，发光器件(LED)发出的光从物体上反射，射入光接收元件，从而可以检测出有物体靠近。

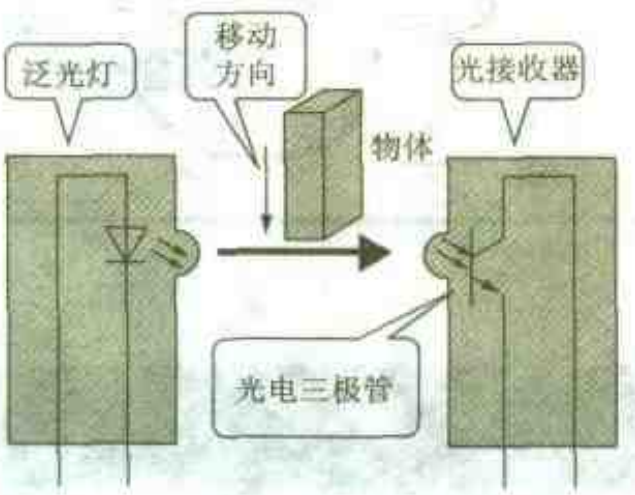


图 2　透射型光电开关

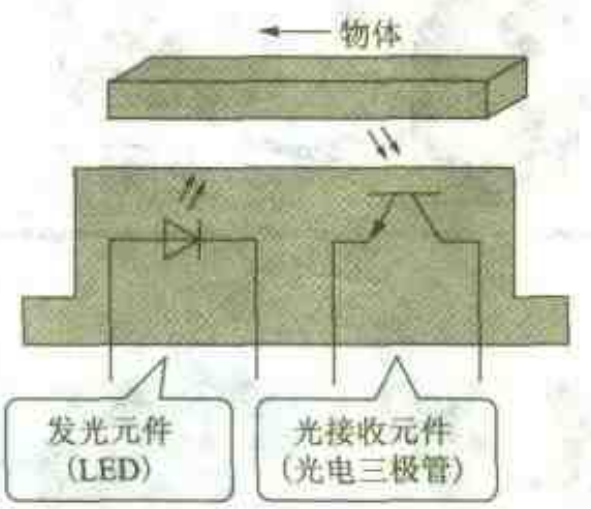


图 3　反射型光电开关

Ⅲ　物体检测电路

图 4 为应用反射型光电开关的物体检测电路的实例。

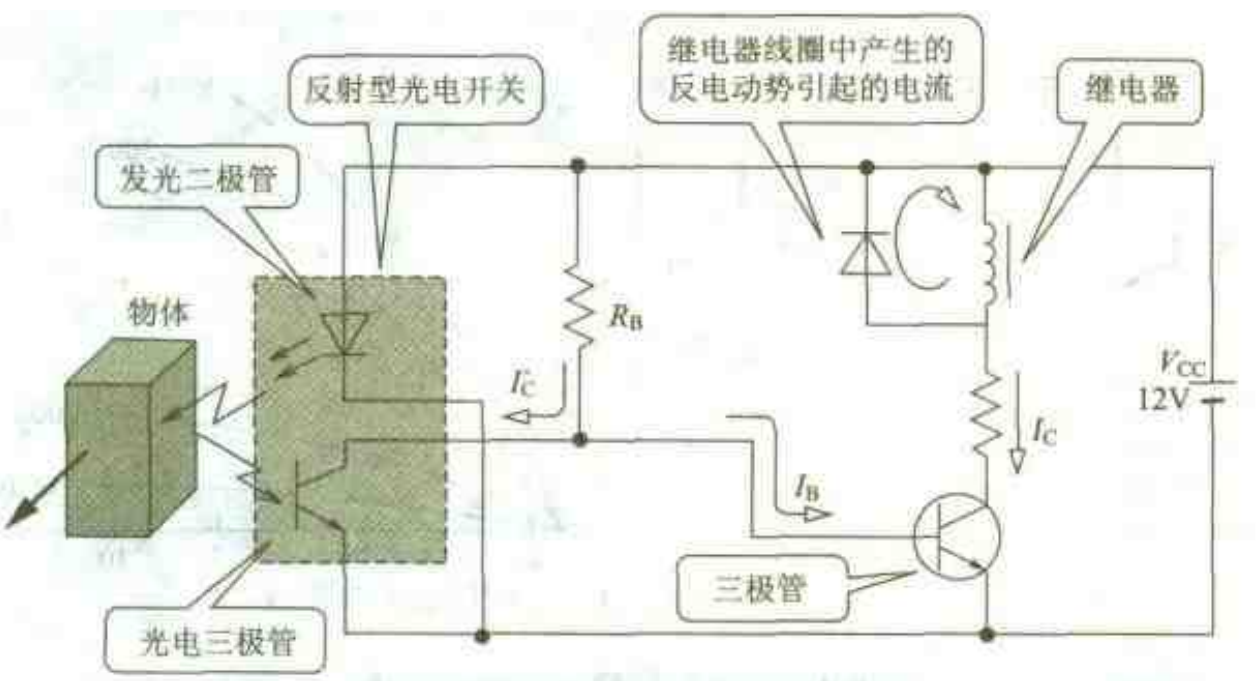


图 4　应用光电开关的物体检测电路

没有物体时，通过 R_B 的电流作为 I_B 流入三极管的基极，流过集电极的电流 $I_C = h_{FE} I_B$，继电器动作。有物体通过时，流过 R_B 的电流作为 I_C' 流入光电三极管，三极管的基极电压随之变得极低，因而三极管截止，继电器不动作。

如此，随着有无物体通过，可使继电器动作或不动作。

Ⅳ　温度传感器

检测温度的传感器称为**温度传感器**。温度传感器有已经学习过的热敏电阻及热电偶。这里介绍测温电阻。

图 5 为铂丝制成的电阻即铂测温电阻的温度特性。如图所示，该特性呈直线变化。图中的 R_0 是 0℃时的电阻，R_t 是温度为 t(℃)时的电阻。

图 6 给出了用铂(或镍)测温电阻测量温度的例子。

测温电阻的电阻值随温度的变化并不非常大，因而如图所示，将其接入惠斯登电桥，将温度的变化变为电压的变化。再将该电压放大去推动温度计的指针，以实现温度的测量。

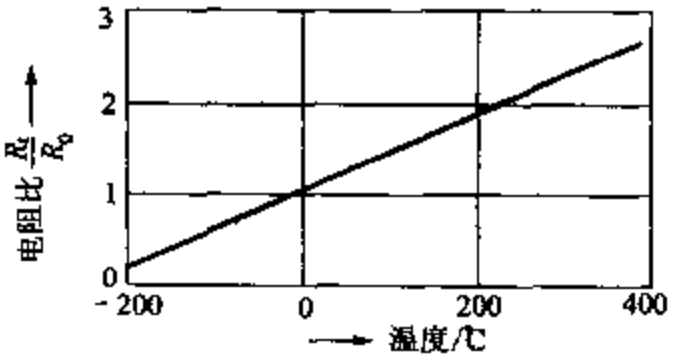


图 5　铂测温电阻的温度特性

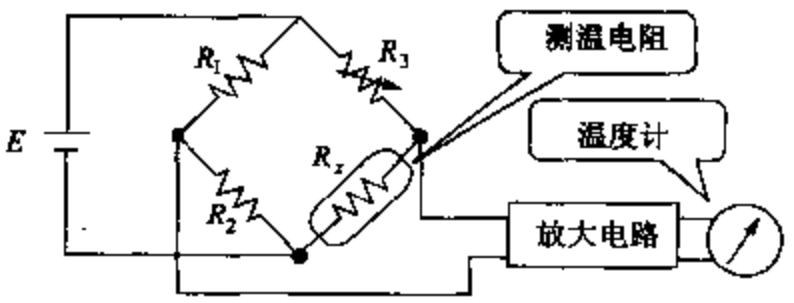


图 6　用测温电阻测量温度

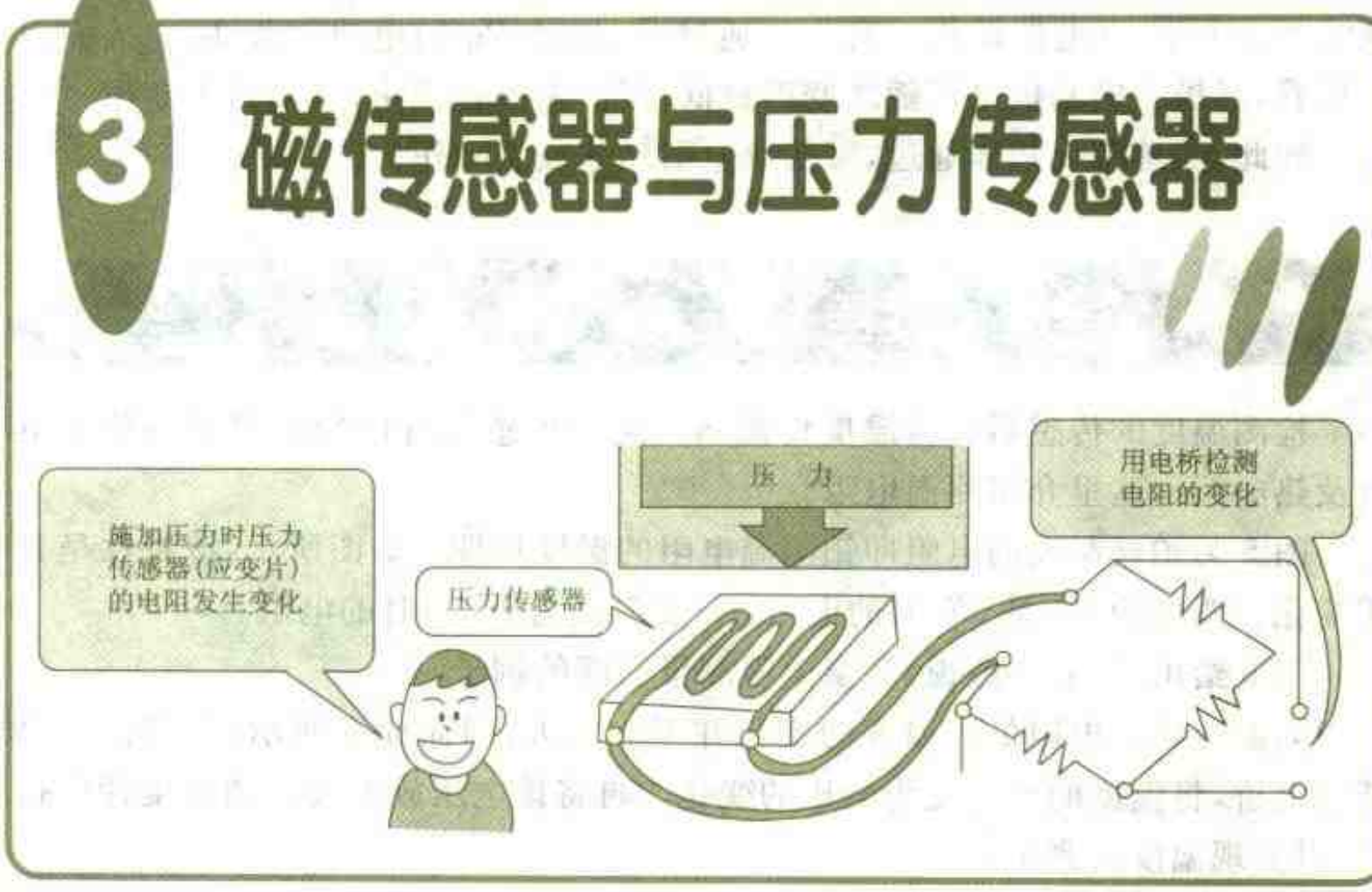
思考题

●将适当的用语填入下面句子的(　)中，使句子正确。

(1) 光电开关可分类为(①)和(②)。无论哪一类型都要有(③)元件和(④)元件。

(2) 应用铂测温电阻测量温度的电路中，要有(⑤)和(⑥)。

3 磁传感器与压力传感器



Ⅰ 磁传感器

检测磁的传感器称为**磁传感器**。磁传感器的典型元件，有已经学过的霍尔元件。图 1 为霍尔元件的工作原理图。

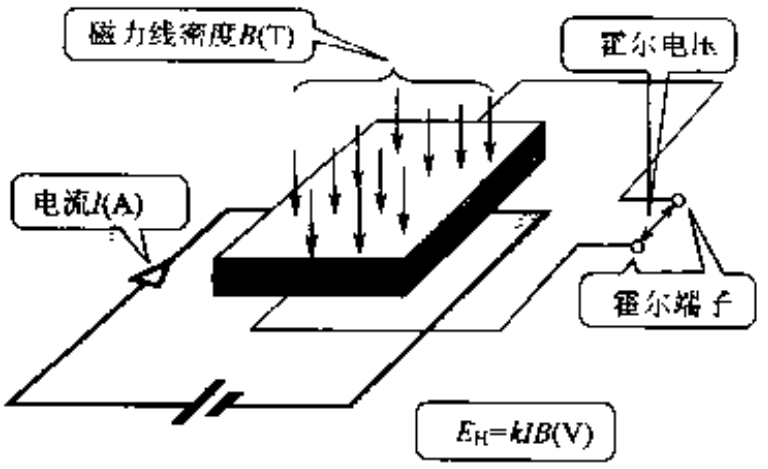


图 1　作为磁传感器的霍尔元件

给半导体通以电流 I(A)，在垂直于此电流的方向上施加磁力线密度为 B(T)的磁场，则在同时垂直于电流和磁场两者的方向上会产生电动势 E_H(V)。

这种现象称为**霍尔效应**，利用此效应可以制作磁传感器。

Ⅱ 舌簧开关

如图 2 所示，舌簧开关的构造为将磁性材料制成的簧片封入细长的玻璃管中。

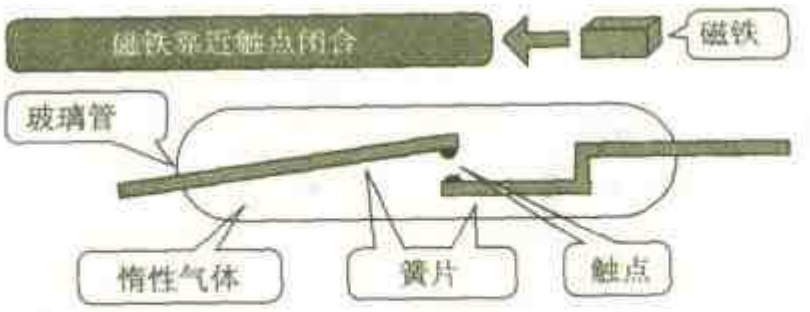


图 2　舌簧开关

有磁铁靠近时，簧片被磁化，簧片头部的触点也被磁化。因而，触点闭合。但是触点开闭时，触点间会稍有放电而使触点氧化。为防止此氧化，玻璃管内充入惰性气体（如氮气、氩气）。

磁铁远离时，由于簧片的弹性，回路断开。这种舌簧开关，外部无磁场时，触点断开（开），外部有磁场时，触点闭合，因而称为**常开型舌簧开关**。

与此相反，外部无磁场时触点闭合（关），外部有磁场时触点断开的开关称为**常闭型舌簧开关**。

舌簧开关的特点归纳如下：

① 触点置于玻璃管中，与外部隔离，因而动作稳定；② 轻小而价格便宜；③ 动作速度快；④ 重复使用精度也好；⑤ 耐电压性能好；⑥ 寿命长。

Ⅲ　压力传感器

图 3 是**应变计型压力传感器**的例子。应变计亦称为**变形计**。

设给金属丝加力拉伸时，其长度由 l 变为 $l+\Delta l$，截面积则由 A 减小为 $A-\Delta A$。此时，长度的增加率称为应变，用下式表示：

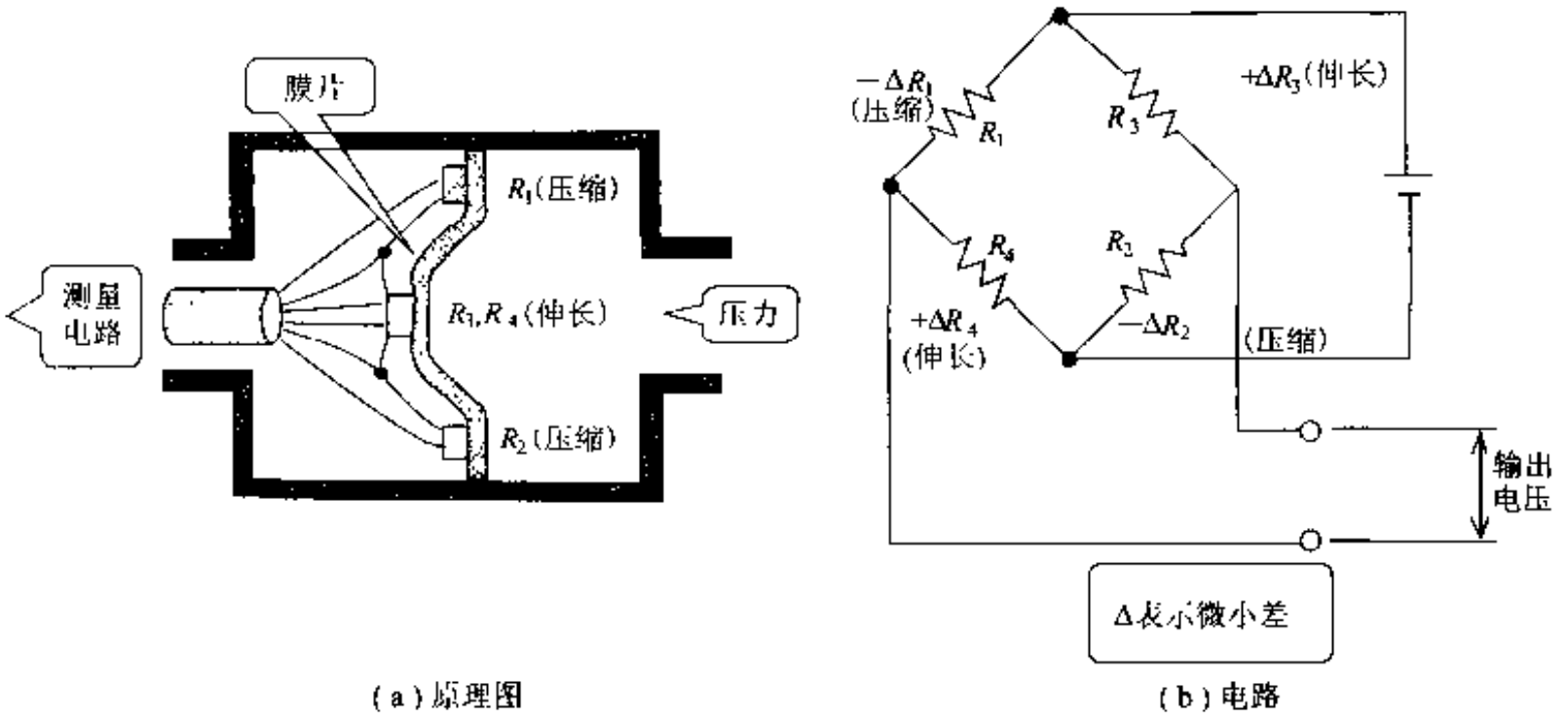
$$应变=\Delta l/l$$

图 3(a)中，施加压力时，贴有电阻应变片的膜片发生变形，因而引起电阻应变片伸缩。其中，电阻 R_1 和 R_2 受压缩，阻值变化为 $-\Delta R_1$ 和 $-\Delta R_2$，电阻 R_3 和 R_4 伸长，阻值变化为 $+\Delta R_3$ 和 $+\Delta R_4$。利用图 3(b)的桥路进行检测，结果以电压形式输出。

设加压造成的电阻值的增加为 ΔR，ΔR 与原电阻 R 的比称为检测率，用 K 表示为

$$K=\Delta R/R$$

作为应变计的必要条件是：相对于变形电阻的变化率要大、温度变化对电阻的影响不大，不易生锈等等。



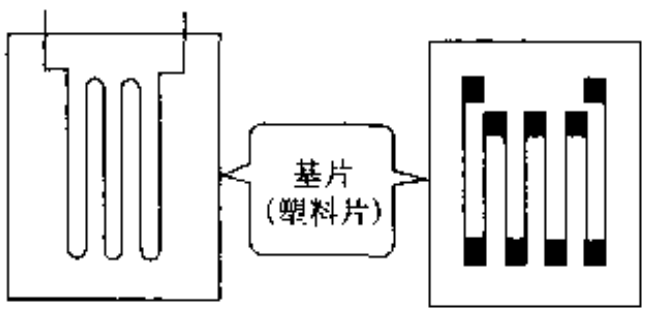
(a)原理图　　(b)电路

图 3　应变计型压力传感器

Ⅳ　应变计

如图 4 所示，应变计有线型与箔型之分。线型是在基片上贴的细金属丝，而箔型是在基片上贴的箔状电阻。

这种电阻的材料，常使用镍铬合金。此应变计的检测率 K 约为 2。



(a)线型　　(b)箔型

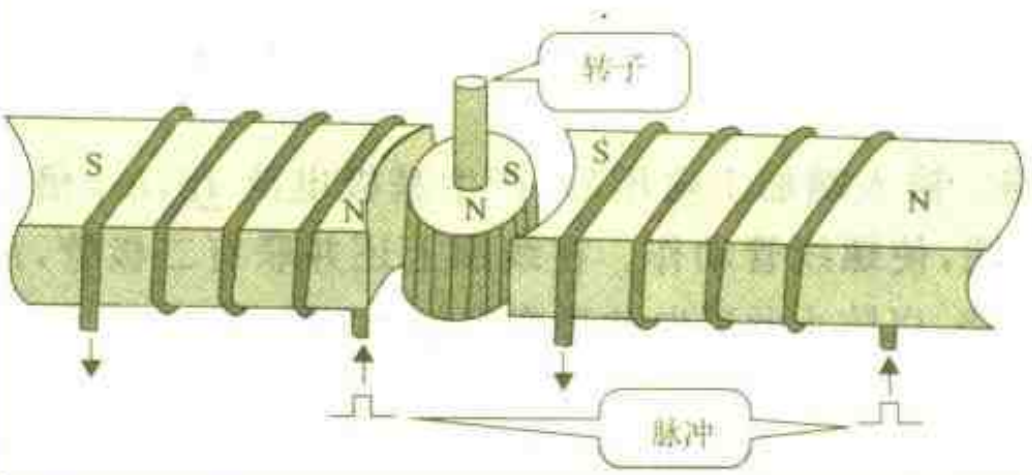
图 4　应变计

思考题

● 将适当的用语填入下面句子的(　)中，使句子正确。

霍尔元件，是(①)传感器，舌簧开关有(②)和(③)两类。检测率 K 用(④)表示，金属应变计的检测率约为(⑤)。

4 电磁螺线管与步进电机





Ⅰ 电磁铁的原理

图 1 表示电磁铁的原理。如图 1 所示，线圈绕在铁芯上，给线圈通以电流，铁芯即被磁化，即可吸引衔铁。衔铁用强磁性材料制作。线圈中无电流流过时，由于复位弹簧的作用衔铁会回到原来位置。这种原理的电磁铁，可用来制作继电器及电磁螺线管。

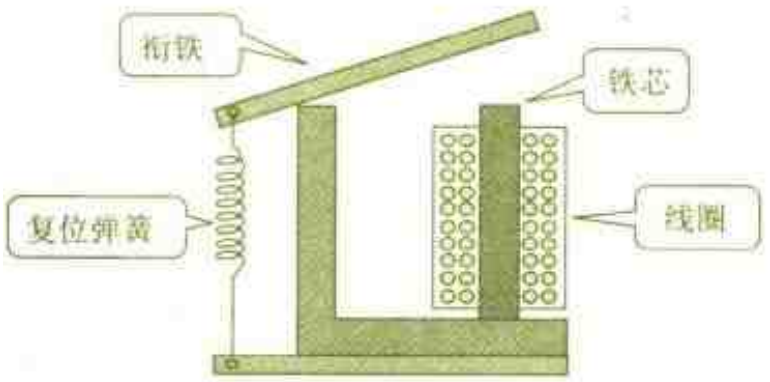


图 1　电磁铁的原理

Ⅱ 电磁螺线管的原理

利用电磁铁的原理，将吸引衔铁或铁片的机械运动用于各种操作的装置，称为**电磁螺线管**或简称**螺线管**。

图 2 为电磁螺线管的原理图。电流流过线圈时，线圈架中的强磁体被吸入架子的中心部。其结果，在左端部产生拉力，而在右端部则产生压力。利用拉力者称为**拉型**，利用压力者则称为**推型**。两者都利用的也有，称之为**两用型**。

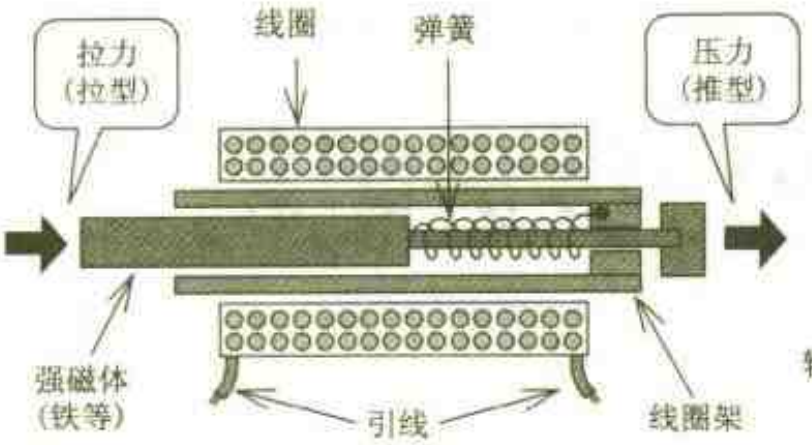


图 2　电磁螺线管的原理

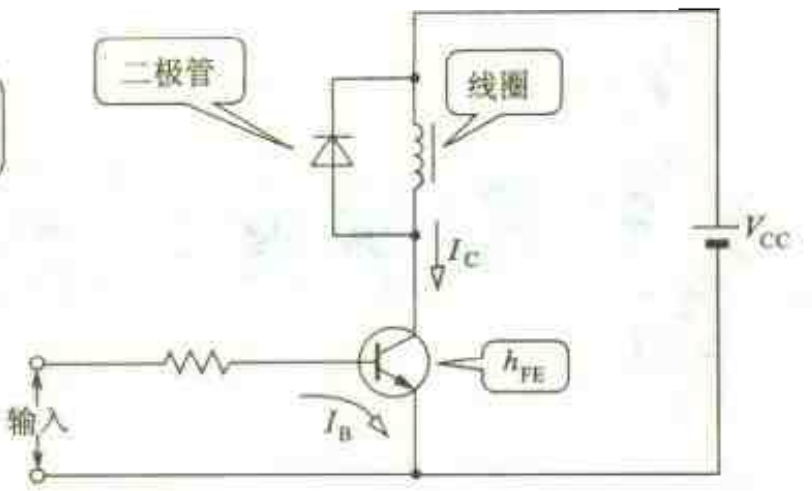


图 3　螺线管驱动电路

图 3 是螺线管驱动电路。输入端加上电压后，产生基极电流 I_B，h_{FE} 倍于 I_B 的集电极电流 I_C 流经线圈，使螺线管动作。在线圈上还并联有二极管，用来吸收线圈上产生的反电动势，以防止此反电动势直接加于三极管上。

Ⅲ　步进电机的构造

图 4 示出了步进电机的构造。如图所示，位于中心称为**转子**的回转部分是永久磁铁。转子周围有称为**定子**的固定部分，定子线圈配置其上。

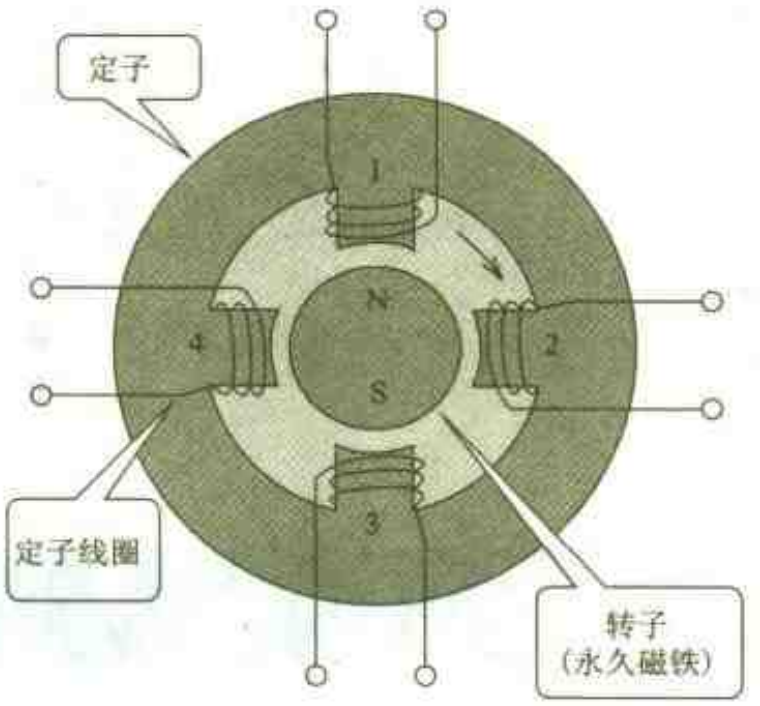


图 4　步进电机的构造

设现在的状态为转子的 N 极正对定子线圈 1，则有以下的动作：

① 电流一流过线圈 2，受 2 号磁极吸引，转子顺时针转 90°；

② 线圈 2 断电，电流一流过线圈 3，受 3 号磁极吸引，转子继续旋转；

③ 同样，顺序给线圈供电，转子沿 4→1→2→3 的方向旋转。

以上说明中虽然讲的是“电流一流过线圈”，但实际上加的是脉冲状的电流。一个脉冲使得回转一定角度的电动机称为**步进电机**。

一个脉冲所旋转的角度称为**步距角**。例如，给步距角为 1.8°的步进电机加 200 个脉冲，如下式所示，转子旋转一圈。

$$1.8^\circ \times 200 = 360^\circ$$

Ⅳ　步进电机的电路构成

图 5 示出了步进电机的电路构成。其中的控制电路用以控制电机的回转方向及回转速度，脉冲发生电路产生按照来自控制电路的控制信号顺序分配给各线圈的脉冲。

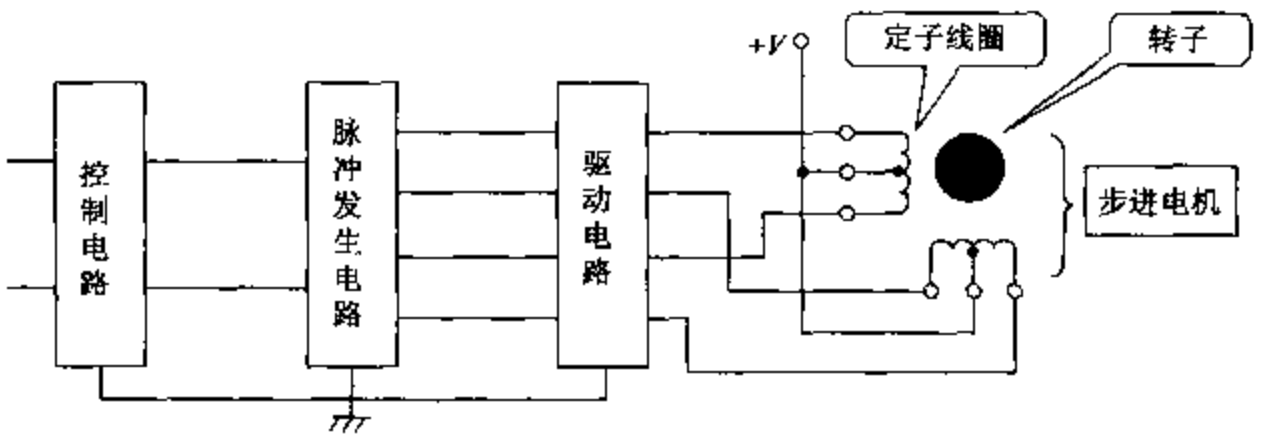


图 5　步进电机的电路构成

驱动电路对脉冲幅度进行放大以供给线圈充分的电流，使脉冲电流流过各线圈。

思考题

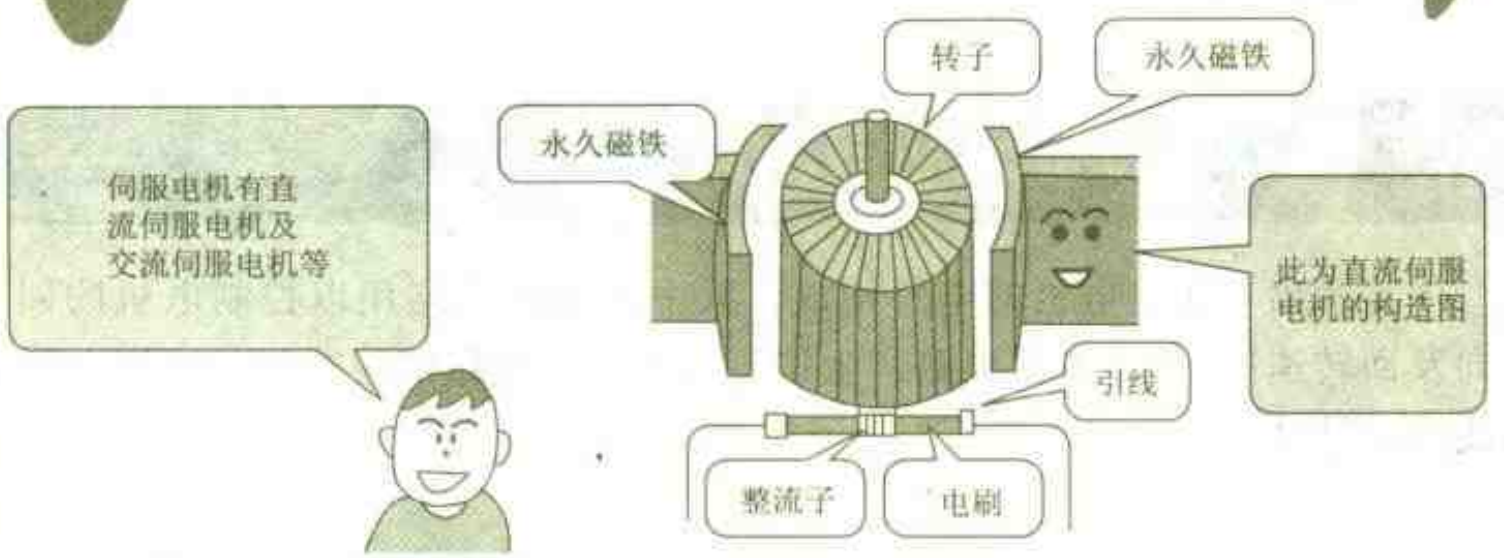
1. 将适当的用语填入下面句子的(　)中，使句子正确。

(1) 电磁螺线管分为(①)型、(②)型、(③)型。

(2) 一个脉冲使得旋转一定角度的电机称为(④)。

2. 给步距角 1.8°的步进电机加 80 个脉冲时，求电机的旋转角度。

5 伺服电机及其应用



Ⅰ 伺服电机

反馈控制旋转速度及机械位置，称为**伺服控制**。所谓反馈控制，是指将控制量与目标值比较，自动地调节控制量的控制。用于伺服控制的电动机称为**伺服电机**。

直流电动机用作伺服电机的称为**直流伺服电机**，交流电动机用作伺服电机的称为**交流伺服电机**。

两者的特点如表1所示。

表1

直流伺服电机	交流伺服电机
①控制装置简单	①故障少
②正转反转简单	②可耐一定程度过载
③速度控制容易	③价格便宜

Ⅱ 直流伺服电机的速度控制

为了对直流伺服电机的速度进行控制，需将**速度发生器**安装于电机上。速度发生器亦称**测速发电机**，它是发电机的一种，用于进行速度检测。

图1为直流伺服电机速度控制电路的构成例。测速发电机安装于伺服电机上，产生正比于电机转速的直流电压。

基准产生与所需转速相对应的设定电压，比较器用来对此设定电压与测速发电机的直流输出电压进行比较。电机驱动电路的任务则是将比较器输出的

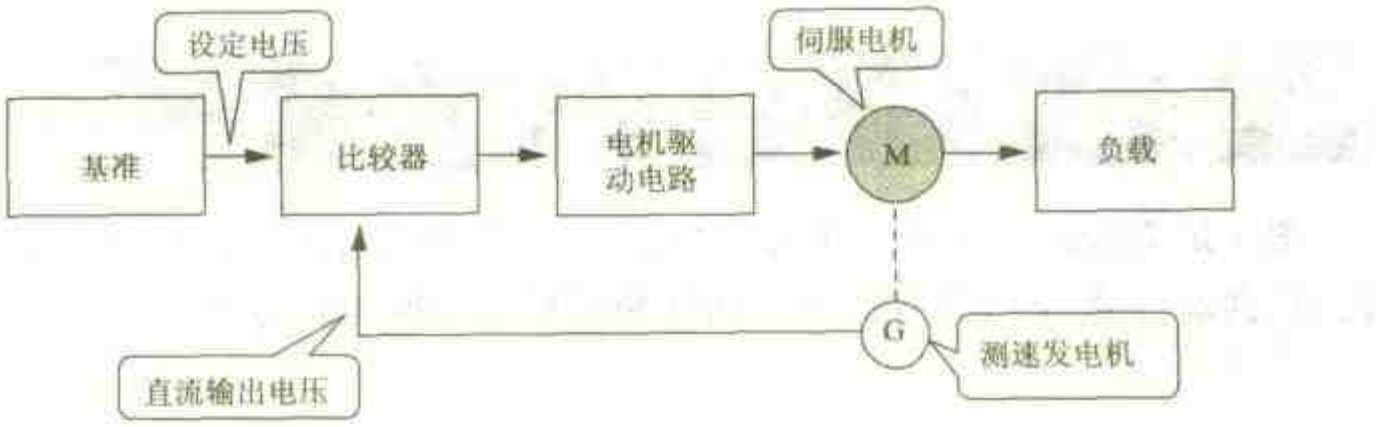


图 1　直流伺服电机速度控制电路

电压送给电机。

现在，假设由于某种原因使得电机的转速降低，则测速发电机的输出电压也会降低→与基准设定电压之差增大→加于伺服电机的电压增大→电机转速得以提高。这是为了使测速发电机的输出电压与基准设定电压趋于相等，伺服电机则以一定的转速旋转。转速的高低可按基准设定电压变化。

直流伺服电机的转速为 1000r/min(每分钟转数)时，测速发电机可产生数伏至数十伏程度的直流电压。

Ⅲ. 定位控制

图 2 为直流伺服电机定位控制的构成例。如果根据输入分压器 R_1 设定目标值，那么电压加在直流放大器的输入上，放大后的电压使直流伺服电机旋转。

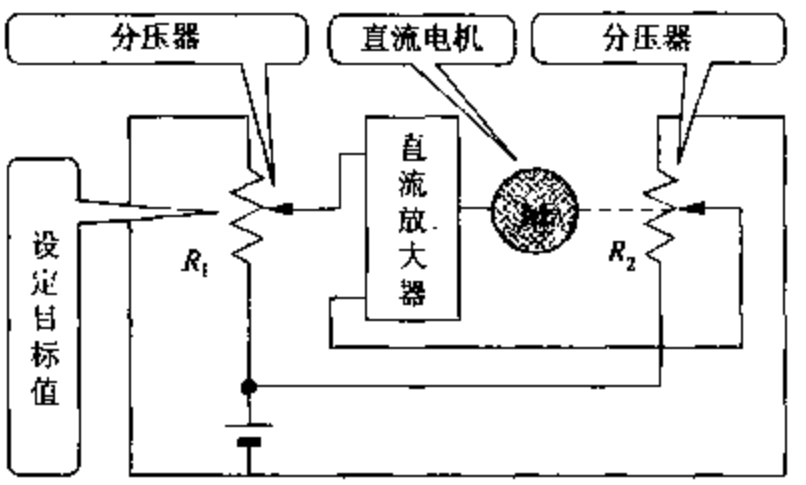


图 2　直流电机定位控制

电机旋转后，来自输出分压器 R_2 的输出电压即被反馈到直流放大器的输入端。反馈值与设定的目标值比较，电机回转直至其差为 0 时停止。这样一来，即可求得正比于输入设定电压的记录。这就是自动记录针的动作原理。

Ⅳ　旋转编码器

图 3 是旋转编码器的原理图。发光二极管的光，通过透光缝射向光电三极管。极据通过缝 A 与缝 B 所产生的脉冲信号可以检测旋转方向。



图 3　旋转编码器

图 4 为内部装有旋转编码器的伺服电机的构成例。将指示旋转角度的指令脉冲输入到比较电路后电机即开始旋转。来自旋转编码器 R 的脉冲与指令脉冲在比较电路中比较，直至其差为 0，电机停止旋转。从而得到所希望的旋转角度。比较电路亦称偏差计数器，这种伺服电机有时称为旋转编码器内置型伺服电机。

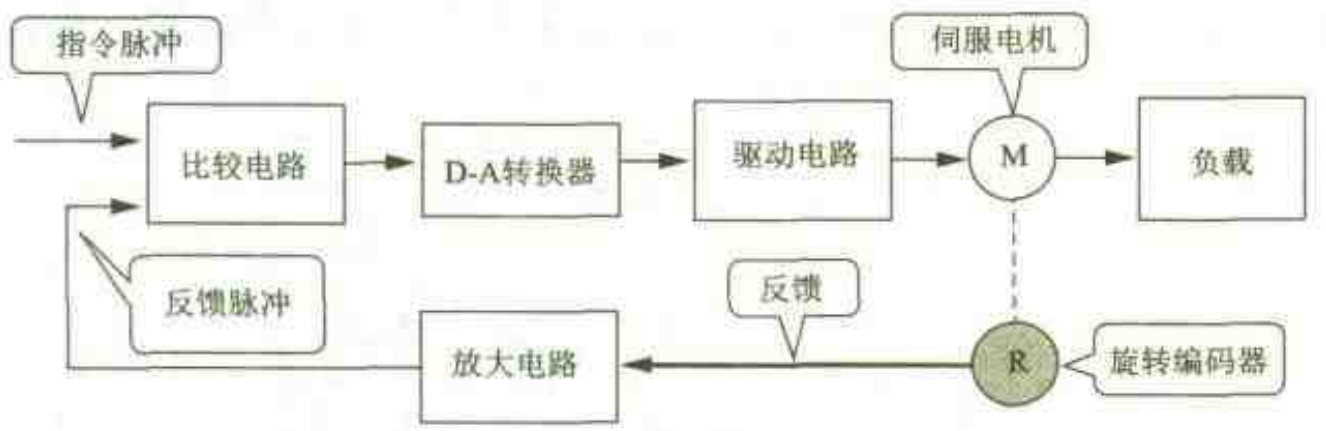
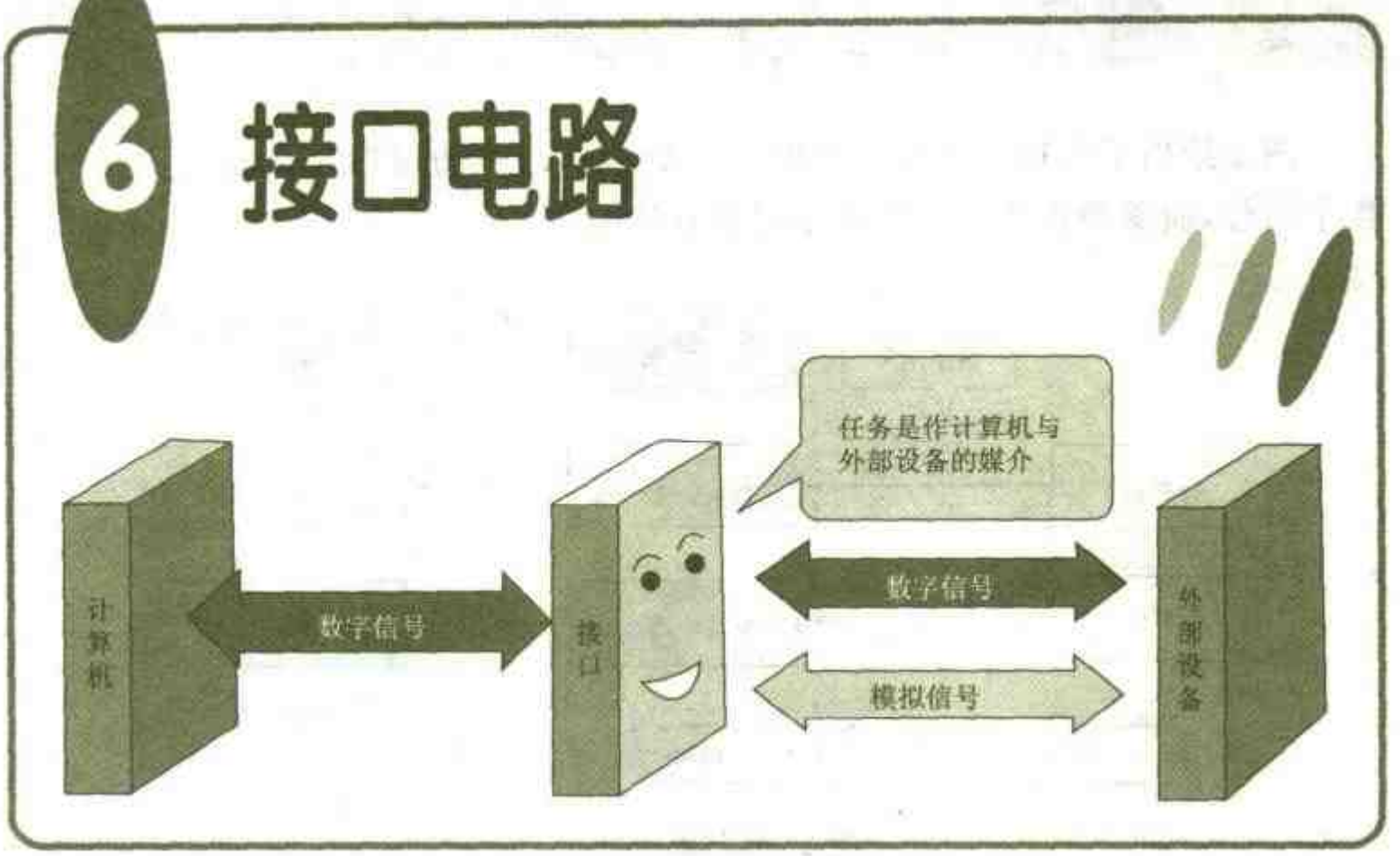


图 4　带有旋转编码器的伺服电机

思考题

1. 试述直流伺服电机的特点。
2. 进行直流伺服电机速度控制时，电机的转速用什么检测？
3. 试述旋转编码器中所用的发光元件与接收元件的名称。

6 接口电路



I 接 口

接口是指介于两个以上的装置之间用于帮助信号传送的装置。图 1 示出了接口的作用。

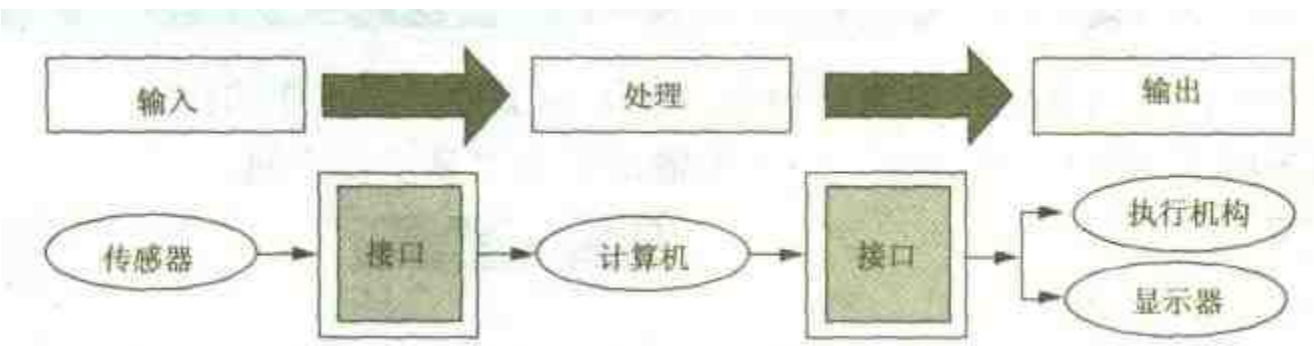


图 1 接口的作用

从传感器得到的信号，原封不动地输入给计算机是不行的。还有，经过计算机处理过的输出信号，原封不动地输入到执行机构或数字显示器也得不到正确的动作。因而，要在传感器和计算机之间、计算机和输出装置之间设置接口，以期得到所希望的信号。

Ⅱ　执行机构的接口

图 2 给出了为执行机构动作所必需的接口任务的框图。计算机输出的是数字信号，而驱动直流伺服电机则需要直流电压。

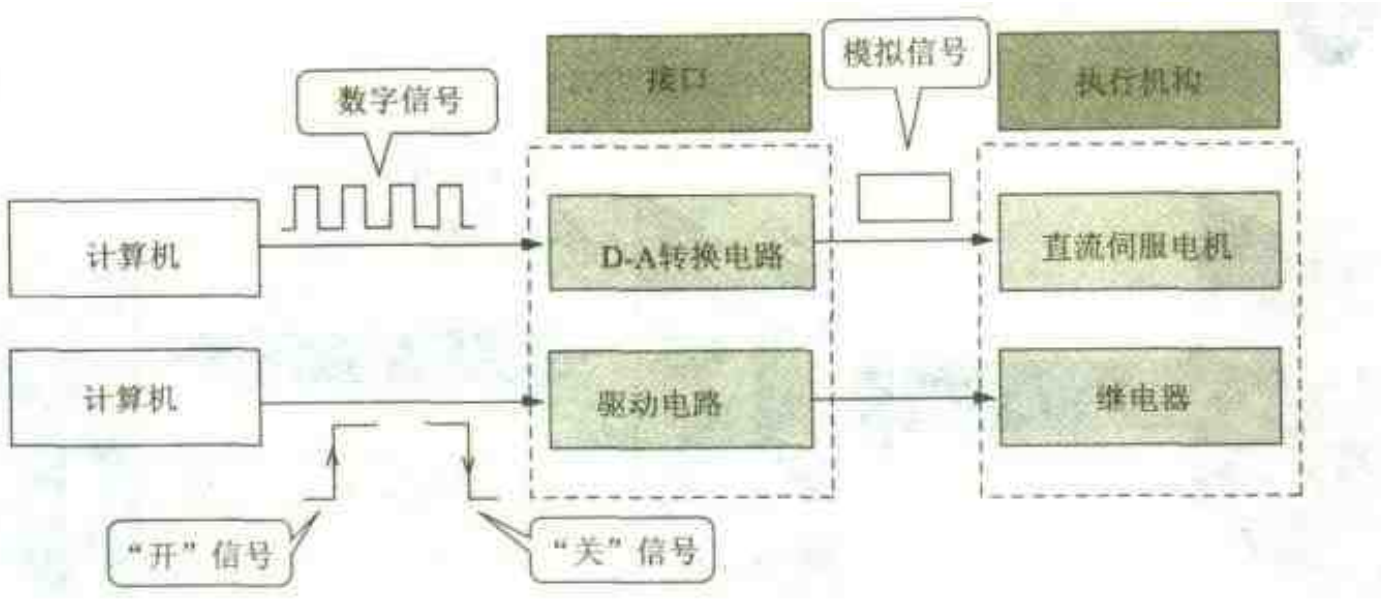


图 2　执行机构的接口

因而出现了将数字信号变为模拟信号的 D-A 转换电路。

还有，要按计算机指令驱动继电器，还必须要有以计算机输出信号为基础的继电器驱动电路。此驱动电路以及 D-A 转换电路都是接口。例如，计算机（微机）的输出电压为 5V，将其变为驱动外设所必需的电压也是接口的任务。

Ⅲ　数字显示器的接口

图 3 示出了为使 7 段发光二极管（LED）数字显示器工作的接口。

如图所示，接口接在计算机与作为输出装置的显示器之间。

二进制数	十进制数
0 0 0 0	0
0 0 0 1	1
0 0 1 0	2
0 0 1 1	3
0 1 0 0	4
0 1 0 1	5
0 1 1 0	6
0 1 1 1	7
1 0 0 0	8
1 0 0 1	9

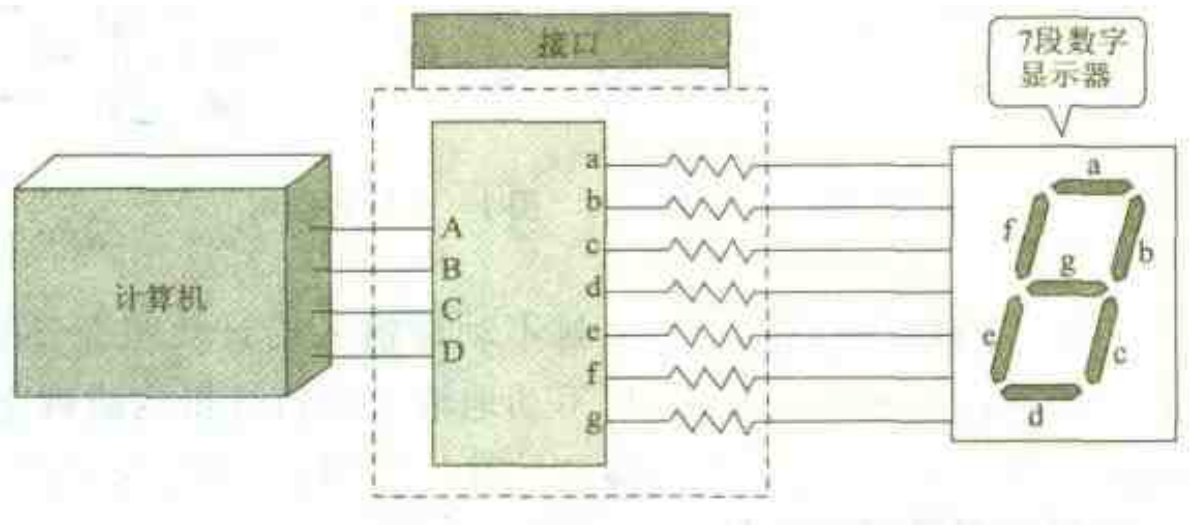


图 3　数字显示器的接口

计算机输出的4位二进制数，要用与二进制数0000～1001相对应的十进制数0～9来表示。

这种将计算机输出的数字信号变成数字显示器(作为执行机构)的驱动信号的装置即是接口。

Ⅳ 输入端接口

机械的触点在通和断时，触点间会在反复多次开闭后，开关才呈现开或闭的状态。这种触点反复开闭的现象称为**抖动**。此抖动会成为计算机测量与控制中产生误动作的根源，因此必须要去除。

图4所示的是用限位开关来检测工作物靠近的工作状态。由于限位开关是机械触点，就会产生抖动。因此用电阻R_1，R_2及电容C接成图示电路，即构成消除抖动电路。

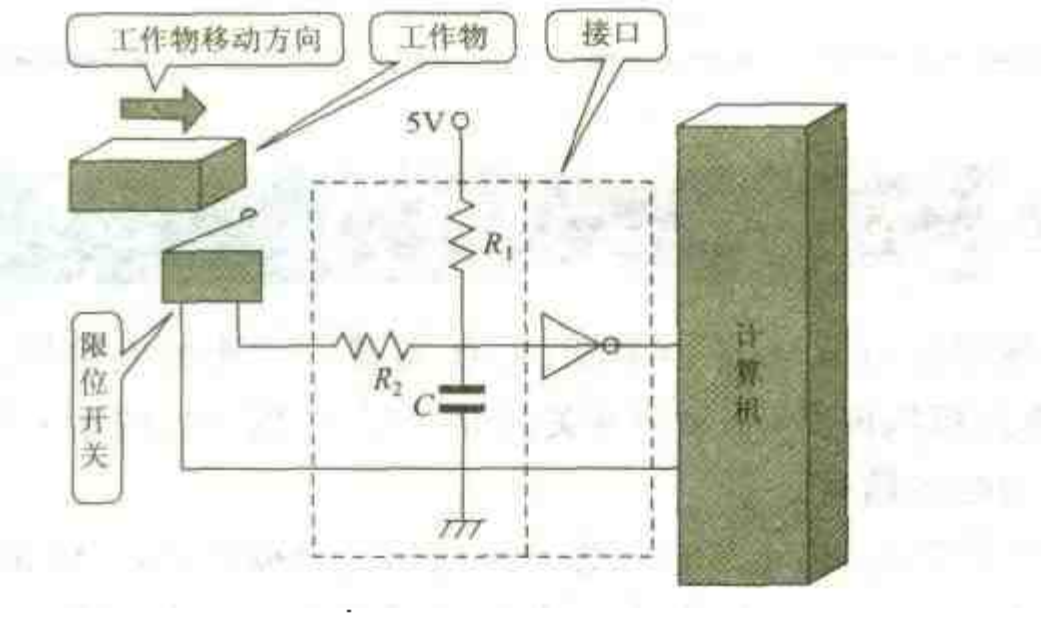
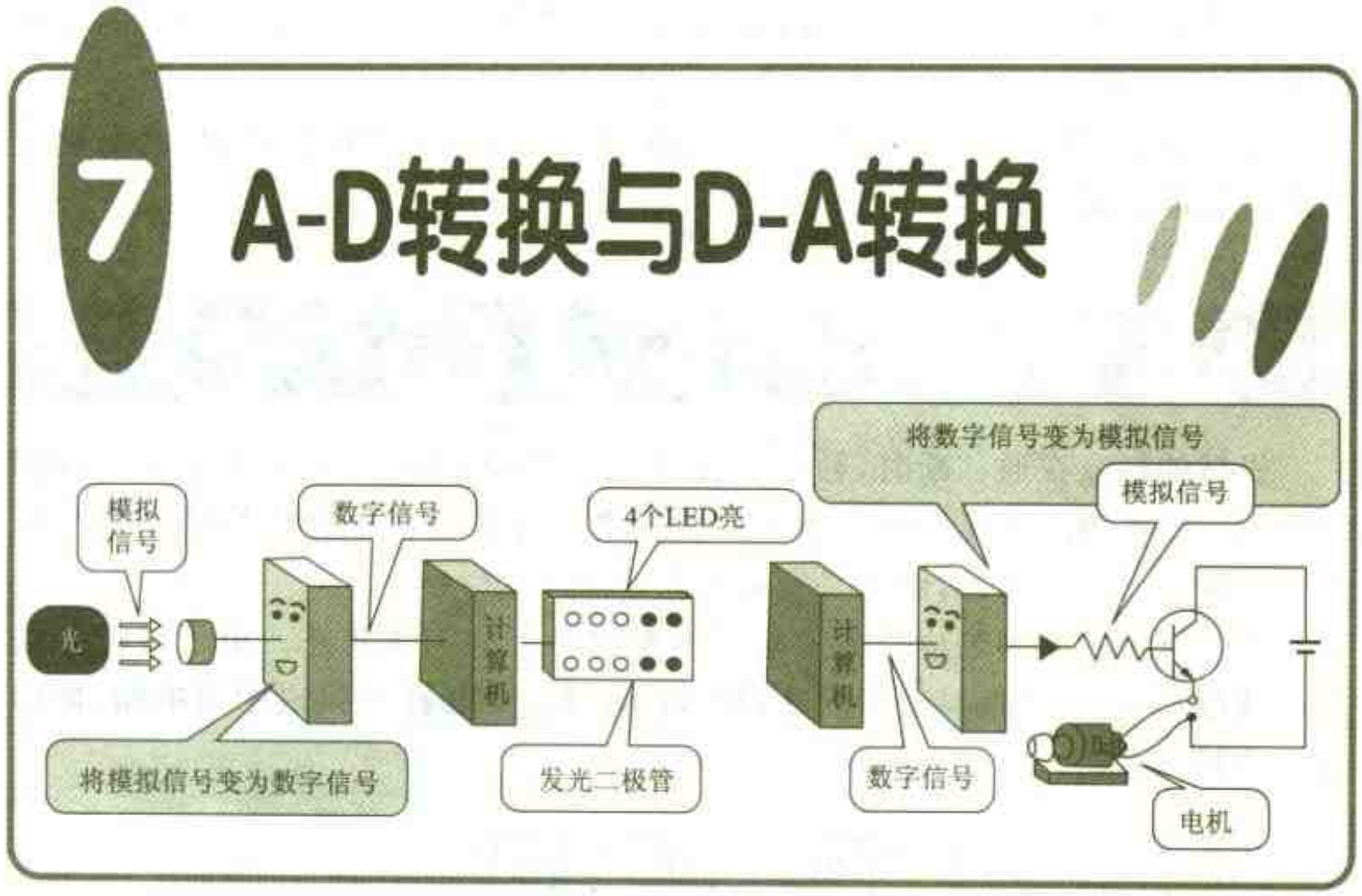


图4 消除抖动电路

图4中的非门是为了开关断开时给计算机的输入为0，而开关闭合时输入为1。所构成的接口包含非门电路在内。

思考题

1. 试述传感器与计算机、计算机与执行机构间所接的电路的名称。
2. 机械触点反复开闭的现象称为什么？



I　A-D 转换

传感器得到的信号，是由温度、压力、亮度等变来的电信号。将连续变化的电信号，称为**模拟信号**；而如同开关的开闭那样，以一定的量为单位阶梯变化的信号则称为**数字信号**。

计算机控制时，由于来自传感器的信号为模拟信号，无法推动计算机工作。因而必须要有将模拟信号变为数字信号的装置。此装置称为 **A-D**(analog-to-digital)**转换器**。图 1 所示的是输入 A-D 转换器的模拟信号转换成二进制数字信号的示意图。

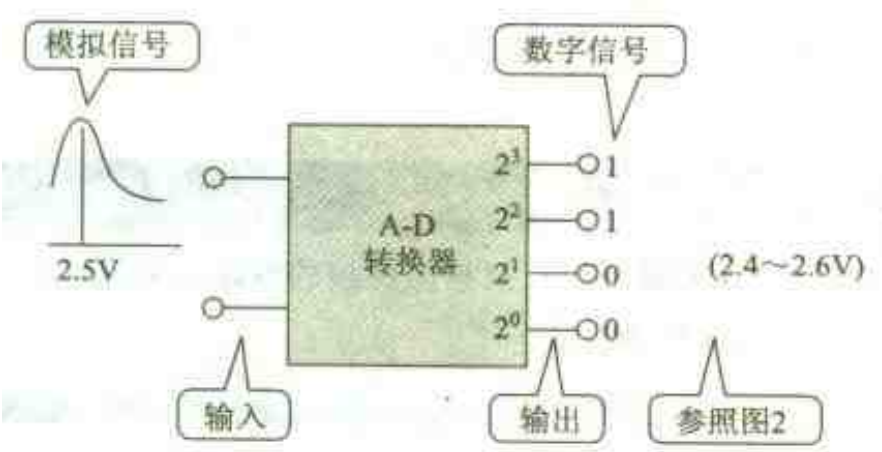


图 1　A-D 转换的方案

Ⅱ A-D 转换例

图 2 给出了 A-D 转换的例子。在这里考虑一下模拟信号在 0～3V 范围内变化时将其变为 4 位数字信号的转换。4 位二进制数的范围是 0000～1111，相应于十进制数的 0～15。因而用 15 去除 3V 得 0.2V。因此，如图 2 所示，每过 0.2V 变一个数字量即可。例如，图 1 中模拟信号的 2.5V 在 2.4～2.6 之间，则将其变为二进制数 1100 的数字信号。由此例可以明显看出，只用 4 位变换的范围过大，因而有时有必要增加位数。A-D 转换器有种种形式，变换的位数也是 4～16 都有，这种转换器已经 IC 化。A-D 转换器多与运算放大器组合使用。

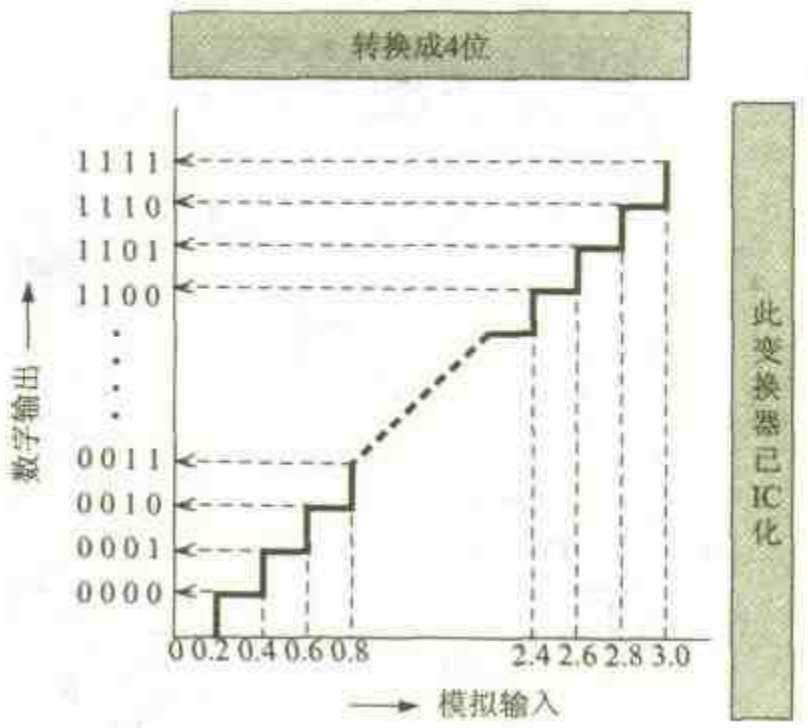


图 2 A-D 转换例

Ⅲ D-A 转换

将数字信号变换为模拟信号，称之为 **D-A 转换**，D-A 转换装置称为 **D-A 转换器**。图 3 示出了 D-A 转换的思路。即在 D-A 转换器的输入端加上相当于 2.4～2.6V 的二进制数 1100 的数字信号时，输出端要出现 2.5V的模拟信号。

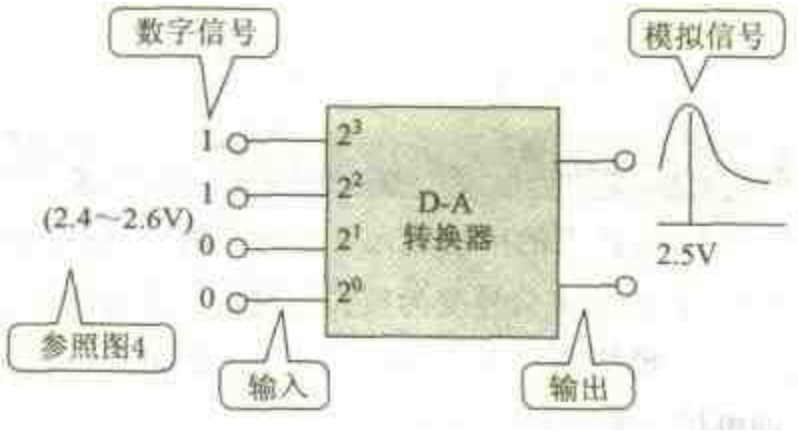


图 3 D-A 转换方案

传感器得到的模拟信号，要经 A-D 转换变为数字信号，计算机才能按此数字信号进行处理。因而，处理的结果是数字信号。此信号无法推动执行机构工作，所以要经 D-A 转换变成模拟信号，才能输入给执行机构。

Ⅳ　D-A 转换例

图 4 为 D-A 转换器的例子。下面考虑将 4 位数字信号变为 0～3V 范围内的模拟信号的转换。4 位二进制数可表示成十进制的 0～15。先来考虑按 4 位变换为 0～3V 范围的模拟信号。3V÷15＝0.2V，因而 4 位数字信号可以每隔 0.2V 变为模拟信号。即数字信号值每增加 1，变换后的模拟信号输出电压增加 0.2V。这种转换器也已 IC 化。

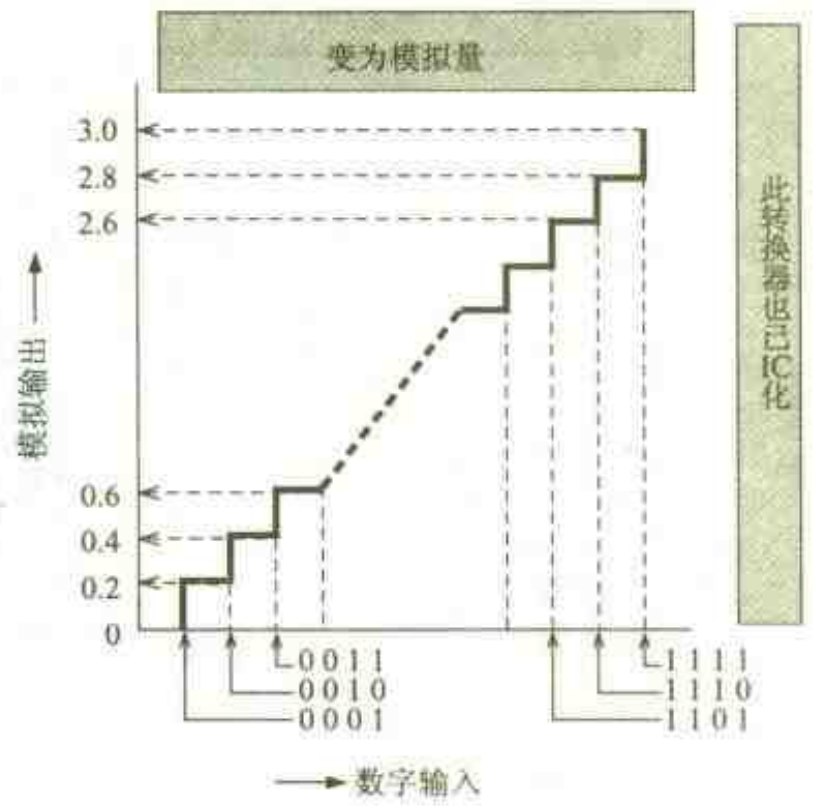


图 4　D-A 转换例

图 4 中，输入数字为 4 位的 1101 时，即十进制的 13，因而模拟输出电压 V_o 为

$$V_o=\frac{3}{15}\times 13=2.6(\mathrm{V})$$

思考题

1. 将适当的用语填入下面句子的(　)中，使句子正确：将模拟信号变为数字信号的装置称为(①)，将数字信号变为模拟信号的装置称为(②)。

2. 模拟电压在 0～5V 范围时，希望用 4 位及 8 位 A-D 转换器将其变为数字信号。求每位的电压。

Q 挑战题

阶梯电路 D-A 转换器虽已经 IC 化，但阶梯电路仍常常使用。

图 1 是梯形电路的原理图。如果图中的数字信号"DCBA"一旦确定，因各位均有一固定的"权"，因而输出电压 V_o(V)为

$$V_o=\left(\frac{1}{2}D+\frac{1}{4}C+\frac{1}{8}B+\frac{1}{16}A\right)(\mathrm{V})$$

$$=\frac{8D+4C+2B+A}{16}(\mathrm{V})$$

如图，设 $D=1, C=0, B=0, A=1, V=5$(V)，则输出电压 V_o 为：

$$V_o=\frac{8\times1+4\times0+2\times0+1}{16}\times5$$

$$=\frac{9}{16}\times5=2.81(\mathrm{V})$$

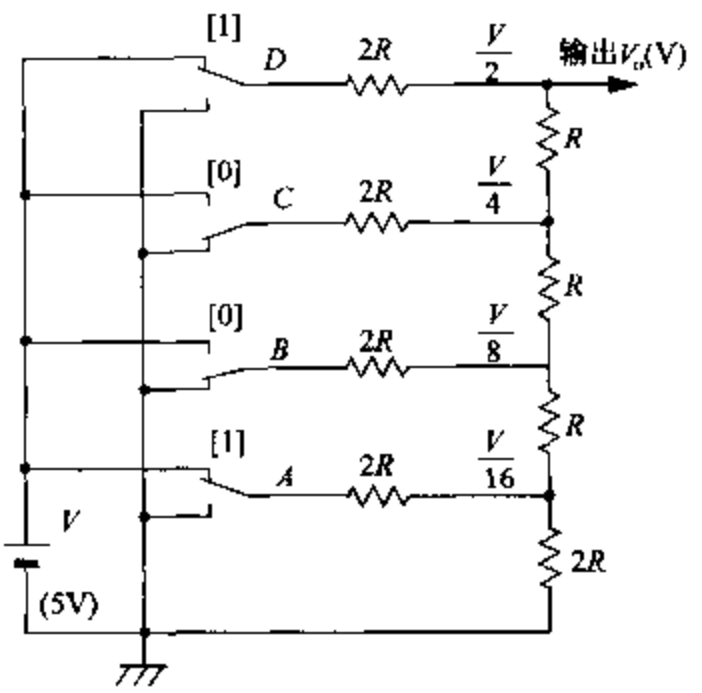


图 1 梯形电路

图 1 使用的是电阻分压电路，称之为**权电阻型 D-A 转换电路**。

之所以称为阶梯电路，是有"梯子"的意思在内，各电阻的端电压像梯子一样，按一定的值上升或说电阻的接法像梯子一样，因而得名。

思考题答案

☞ p. 4

1. ①传感器 ②执行机构 ③电气型 ④气压型 ⑤油压型 ⑥接口

☞ p. 7

1. ①透射型 ②反射型 ③发光 ④接收 ⑤放大电路 ⑥温度计

☞ p. 10

1. ①磁 ②常开型 ③常闭型 ④$\Delta R/R$ ⑤2

☞ p. 13

1. ①拉、②推、③两用 ④步进电机
2. 144°

☞ p. 16

1. 控制装置简单，正转反转简单，速度控制容易。
2. 测速发电机
3. 发光二极管、光电三极管

☞ p. 19

1. 接口
2. 抖动

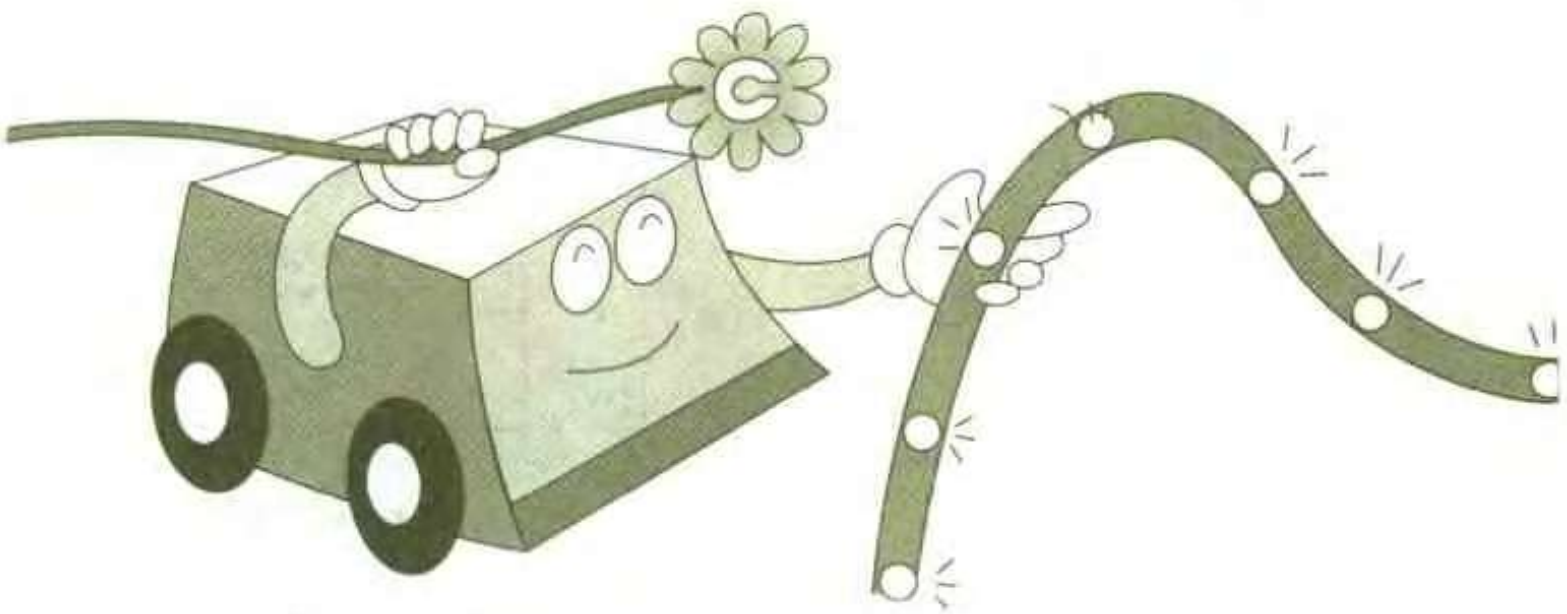
☞ p. 22

1. ①A-D 转换器 ②D-A 转换器
2. $5\div(2^4-1)=0.333(\mathrm{V})$

 $5\div(2^8-1)=0.0196(\mathrm{V})$

第2章

C语言控制入门



本章学习内容

袖珍计算机不仅具有计算及BASIC语言功能,还有种种功能。本章通过用C语言编制简单控制程序能体验学习C语言的乐趣。本章将具体讲解:

1 LED的闪烁控制

2 LED闪烁速度的调整与音阶

3 LED闪烁的减速

1 LED的闪烁控制

使用袖珍计算机进行控制时，必须要有将来自袖珍计算机的控制信号传向外部及将数字信号传向袖珍计算机的接口。这里使用采用 PPI-82C55 IC 的接口“宽宽”。“宽宽”是接口板的爱称。在电路控制板上，将 24 个 LED 排成一个圆圈，恰如轮盘赌那样的外形。

闪烁的同时还要产生音阶，因而准备了 8 个按键开关和两个拨钮开关，是一块通过仔细编程能够获得种种体验的控制板。此控制板的爱称为“圆圆”。

“宽宽”和“圆圆”是日本 YOUPIT 电子公司开发的商品。“宽宽”和“圆圆”的关系详细编辑收录在录像带上出售。看着录像进行学习，能够在短时间内获得更好的 C 语言学习效果。

I “圆圆”和“宽宽”的自我介绍

初次见面，我叫“圆圆”。由于一圈一圈地转，有了“圆圆”这个名字。实际上我并不转，只是排成转盘状的 24 个 LED 顺序闪烁而已。但是，大家都叫我“圆圆！圆圆！”，这样不知不觉中就有了“圆圆”这个名字。我也不怎么生气，叫这个名字挺好。

我靠着与袖珍计算机和接口联接而工作。我会忠实地按照袖珍计算机编制的程序工作[1)]。

为使用 C 语言初始编制控制程序的人着想，作为初等生的“圆圆”，先将 8 个 LED 排成一排。

1) “宽宽”是作为接口的爱称而使用。所谓“宽宽”指的是装有一片 PPI-82C55 芯片的袖珍计算机接口。

Ⅱ 我的外观与袖珍计算机的外设

构成"圆圆"的有：呈轮盘状排列的24个LED、处理这些LED输出信号的5个锁存IC芯片及4个晶体管列阵(图1)。

还有一排8个LED及与其并排的8个按钮。这些按钮是输入装置。按下此按钮开关，LED会发光、喇叭会发出声音。

图中两只拨钮开关作为输入开关，用于选择输入，可有6种输入；有两只输入缓冲器IC。

有关接口的更详细的情况以及袖珍计算机的信号关系，在下节说明。

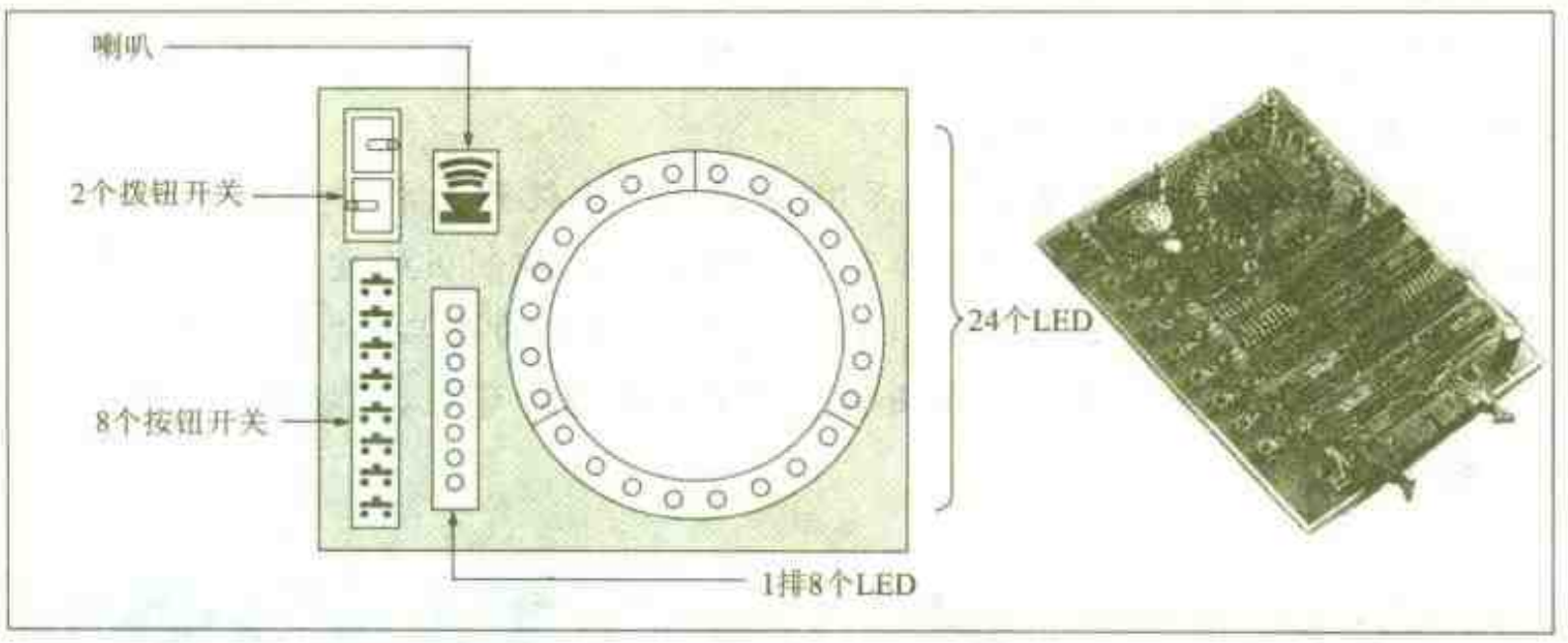


图1 "圆圆"的构成

Ⅲ C语言控制基础(输出篇)

下边使用CASIO VX系列袖珍计算机及Z-1来进行袖珍计算机控制。来自袖珍计算机的控制信号可从以下两处取出：

①30针杜邦制连接器，

②40针插卡连接器。

本书将以从30针杜邦制插口取出信号的方法为中心进行介绍。

(1) 30针扩展引脚的连接信号

Z-1袖珍计算机因是VX系列的改进型，所以其中仍使用30针扩展连接器。

表1中列出的是30针扩展连接器的各针信号。

表1 30针扩展引脚的连接信号

针 号	名 称	输入输出	用 途
1	V_{CC}		+5V
2	PB8	输入	RS232C DSR
3	PB7	输入	RS232C CD
4	PB6	输入	RS232C CTS
5	PB5	输入	未使用
6	PB4	输入	BUSY(PRINTER)
7	PB3	输入	未使用
8	PB2	输入	未使用
9	RXD	输入	RS232C DATA IN
10	PA4	输出	INIT(ACTIVE LOW)
11	PA3	输出	STB(ACTIVE LOW)
12	PA2	输出	RS232C DTR
13	PA1	输出	RS232C RTS
14	TXD	输出	RS232C DATA OUT
15	RST	输出	RESET OUT
16	DB4	输入输出	DATA BUS
17	DB5	输入输出	DATA BUS
18	DB3	输入输出	DATA BUS
19	DB6	输入输出	DATA BUS
20	DB2	输入输出	DATA BUS
21	DB7	输入输出	DATA BUS
22	BD1	输入输出	DATA BUS
23	DB8	输入输出	DATA BUS
24	A0	输出	ADDRESS BUS BIT 0
25	A1	输出	ADDRESS BUS BIT 1
26	A2	输出	ADDRESS BUS BIT 2
27	OE	输出	OUTPUT ENABLE
28	WE	输出	WRITE ENABLE
29	CS3	输出	CHIP SELECT 3
30	GND		0V



从此 30 针扩展连接器取出的信号可用于外部机器的控制及通信。从此连接器取出信号作成的接口板即为“宽宽”C 系列或就叫“宽宽”。图 2 是“宽宽”的电路图。

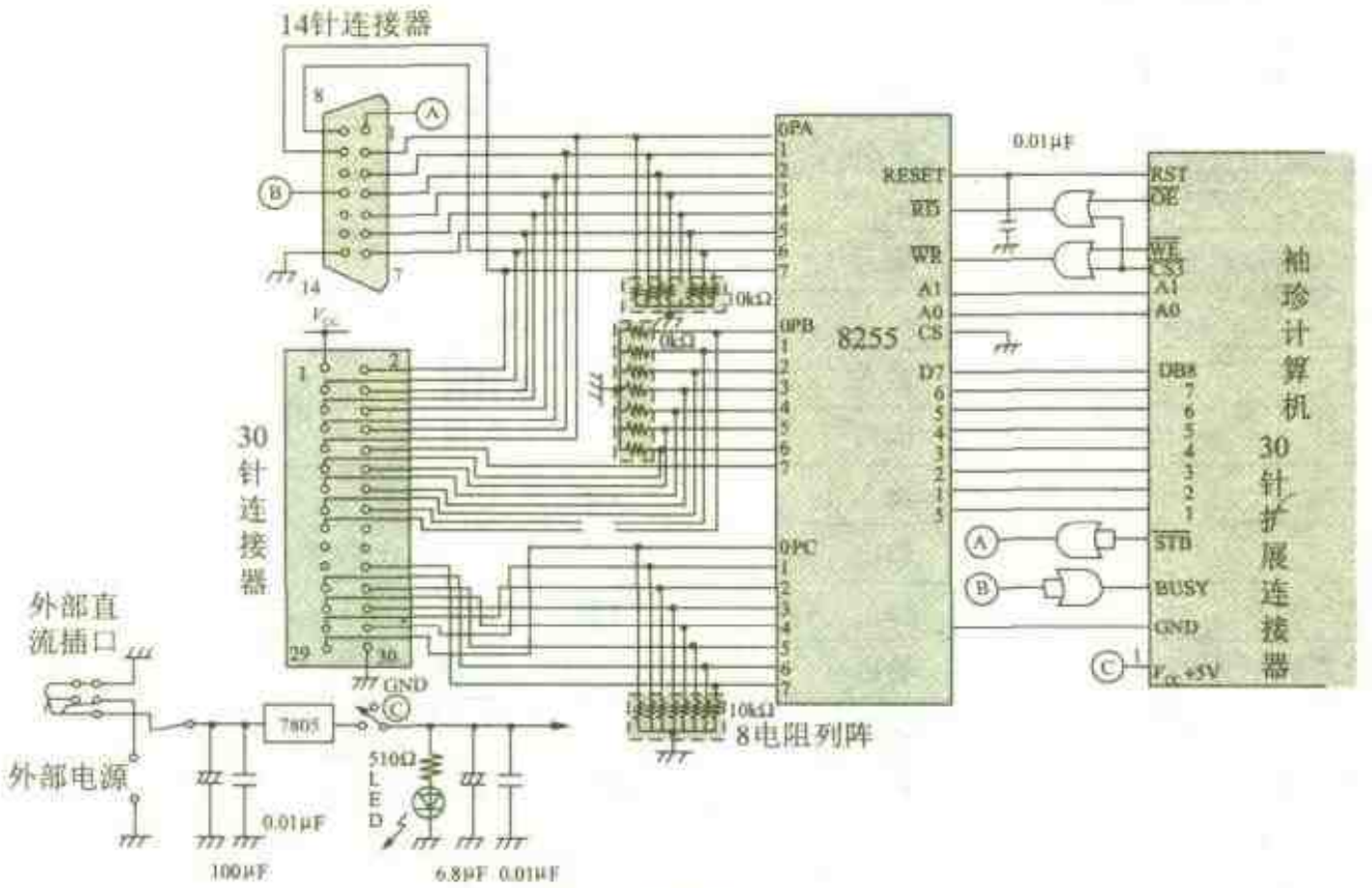


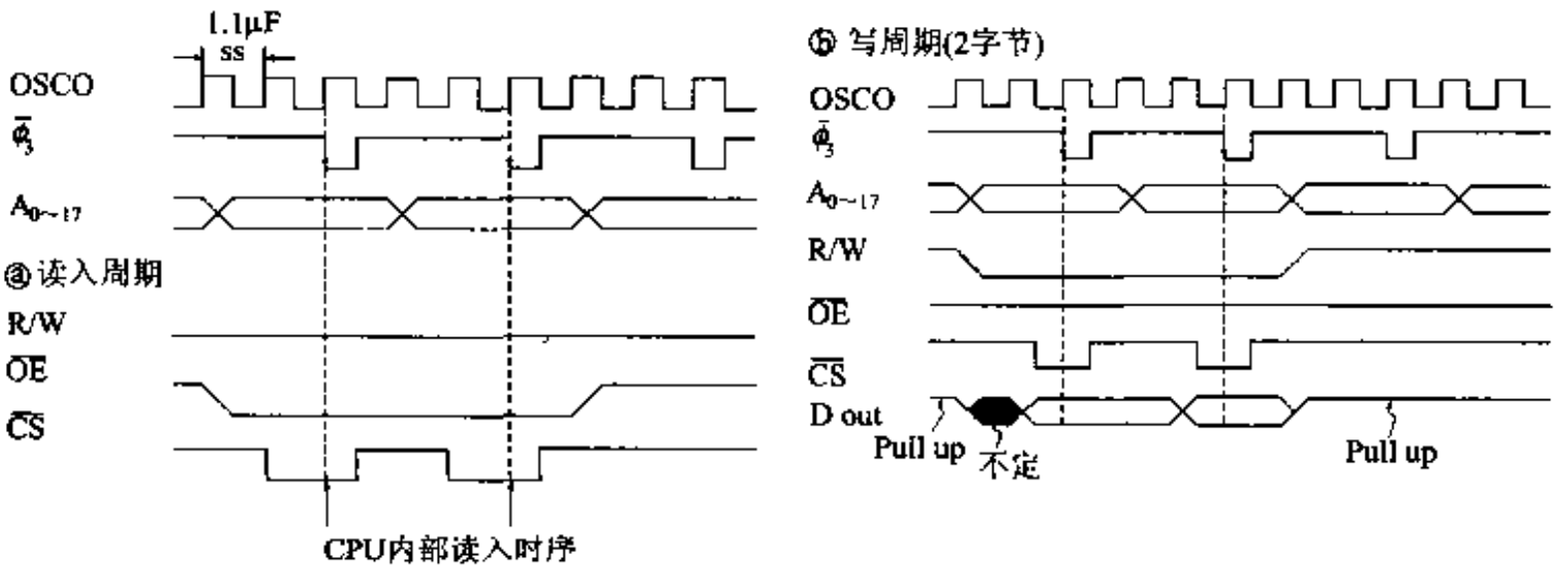
图 2　“宽宽”电路图

袖珍计算机的电源为正电源(低电平为零),即

$V_{CC}=5V\pm2.5\%$。

作为控制总线,可使用 CS3,OE,WE。地址线只能使用低 3 位(A0～A2)。数据总线为 DB1～DB8 8 位。

读写时序请参照图 3(a)、(b)。



(a) 读周期时序　　(b) 写周期时序

图 3　时序的安排

为将82C55接口IC芯片的3个口8位(合计24位)信号引向外部,特准备有30针连接器。

如图4所示,电路板的背面为有焊锡的一面。这样,右侧的第30针就是"地(GND)"。此面上涂有绿色的抗蚀剂,可以理解为地是电路板上导通面积最大的。

图5所示为焊锡面。

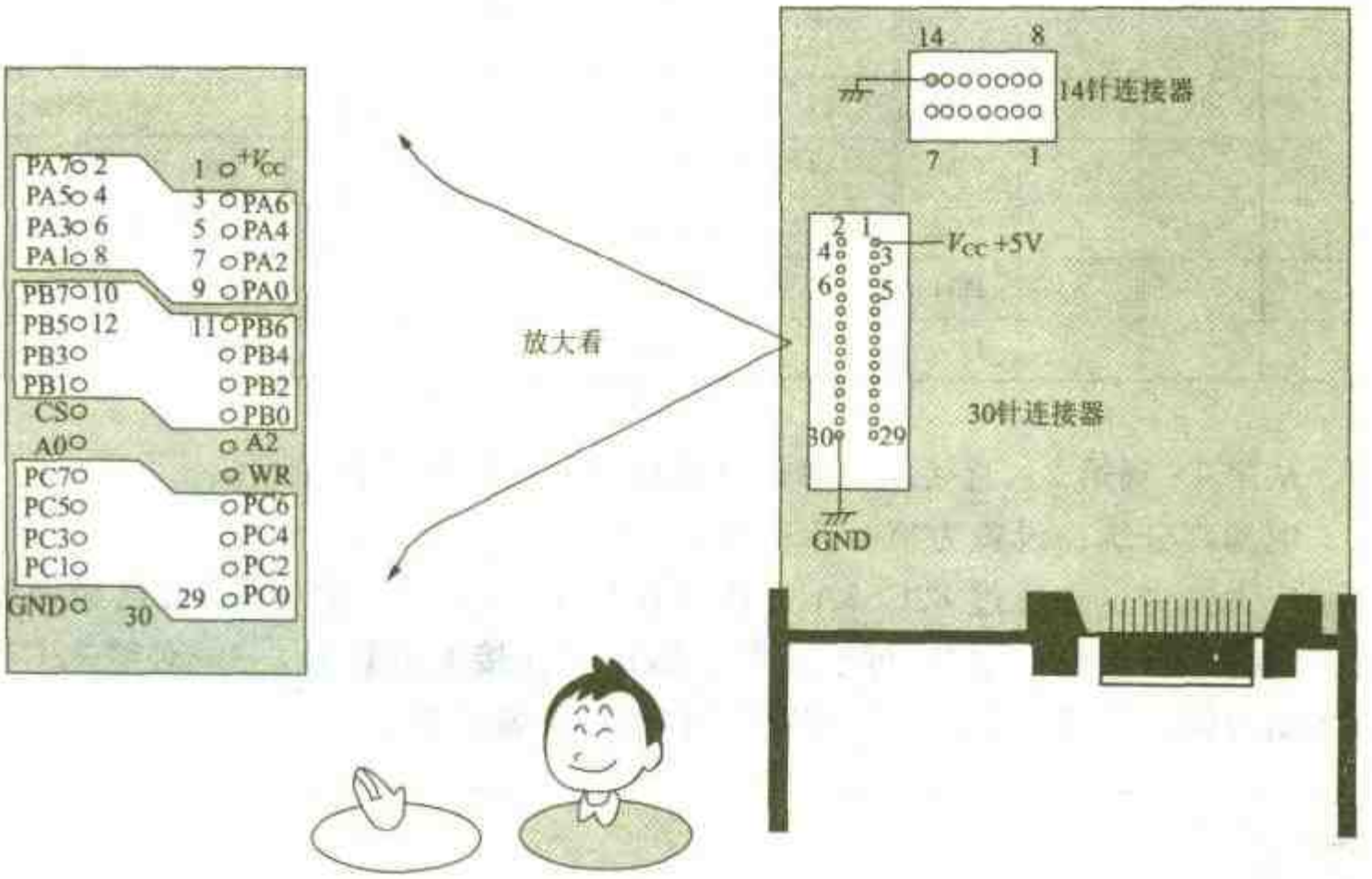


图4 "宽宽"边上连接器针的分配

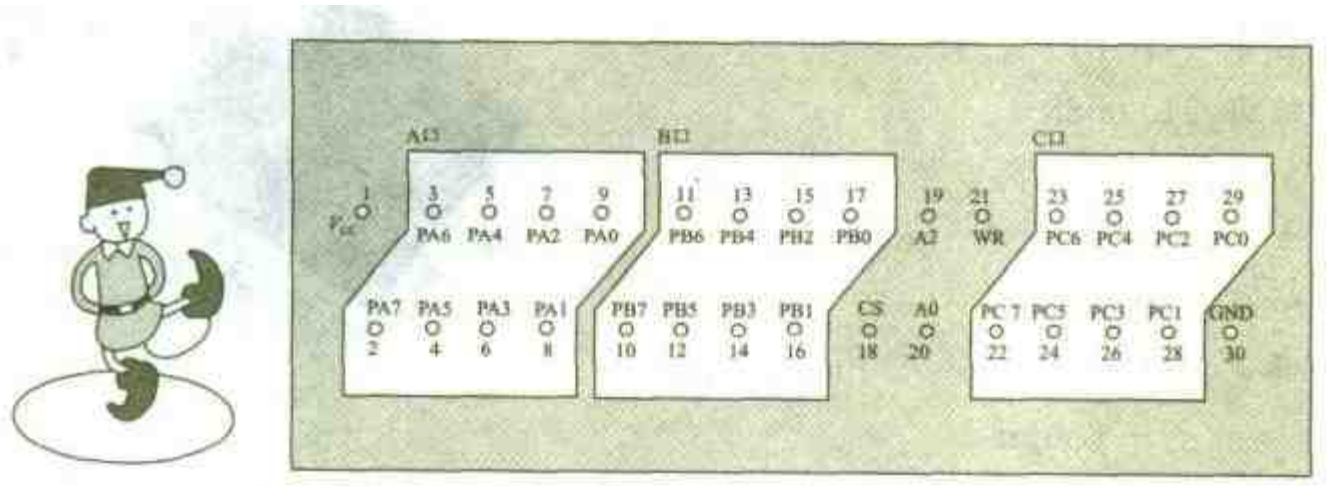


图5 从背面看到的30针连接器的针的序号

表2示出了30针连接器的针的分配。"宽宽"与外部信号交换是通过30

针连接器来进行的。从图 5 也可以看出，82C55A 的 A，B，C 口的信号线都归在了 30 针连接器的各相应口中。A 口是从第 2 针到第 9 针，B 口是从第 10 针到第 17 针，C 口是从第 22 针到第 29 针。

表 2 30 针连接器的针分配

针号	电源	针号	A 口	针号	B 口	针号	控制信号	针号	C 口
1	$+V_{CC}$	2	PA7	10	PB7	18	CS	22	PC7
30	GND	3	PA6	11	PB6	19	A2	23	PC6
		4	PA5	12	PB5	20	A0	24	PC5
		5	PA4	13	PB4	21	WR	25	PC4
		6	PA3	14	PB3			26	PC3
		7	PA2	15	PB2			27	PC2
		8	PA1	16	PB1			28	PC1
		9	PA0	17	PB0			29	PC0

从第 18 到第 21，直接从袖珍计算机取出原始信号以供使用。

电源部分统一设置为第 1 针和第 30 针。

广大读者也许通过 82C55A 进行过控制实验吧！想用“宽宽”对已经做好的控制物进行动作控制时，可参照图 5 给出的连接器的针号，一一对照 82C55 的各信号线。这样一来，使用“宽宽”的控制会非常广泛。



袖珍计算机扩展板的销售公司

你要购买袖珍计算机的扩展板，请咨询。

日本 YOUPIT 电子公司

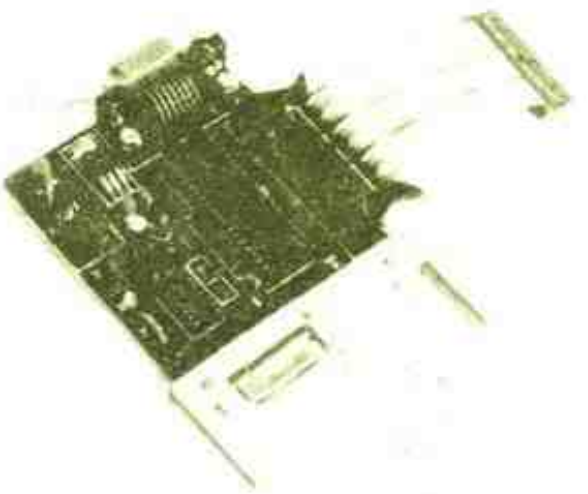
邮编：940-0088

地址：新潟县长冈市柏町 2—2—8

电话： 0258—32—2646

传真： 0258—33—2012

网址： http://www2.hits.ad.jp/~youpit



(2) 40 针插卡连接器

在袖珍计算机 Z-1 中，除了 30 针的扩展连接器外，还增加有 40 针的插卡连接器。由此 40 针插卡连接器也可以取得 PPI 控制接口的信号。这里虽不使

用，但作为参考，也将其规格作一介绍。

40针插卡连接器，输出的是CPU80L 188EB的总线。16位的地址必须译码，但为了使用方便，信号从CS1(芯片选择1)的引线输出。此信号的作用是归纳整理地址的高10位和IORQ(IO请求)信号。

为通过插卡连接器从Z-1袖珍计算机对外部设备进行存取，首先要设定电路片选信号CS1(第4针)和CS2(第5针)的使能条件。

对应于此插卡CS1的CPU内的寄存器为GCS4。同样对应于CS2的寄存器为GCS5。根据这些寄存器来指定是在I/O空间，还是在存储空间让片选使能。

设定例

分配CS1为0300H～03FFH的I/O空间时，

```
OUT &H90,&H0300,2.   设定GCS4开始...设定40针插卡连接器的CS1开始
OUT &H92,&H0408,2.   设定GCS4停止...设定40针插卡连接器的CS1停止
                     地址设定时为2
                     数据设定时为1
```

用C语言编写程序时，写法如下：

```
main()
{
  int i,a,b;
  outport (0x90,0x0300,2);
  outport (0x92,0x0408,2);          } 地址初始设定
  outport(0x323,0x90,1);      ....  设定CWR为0x90
  outport(0x322,0x3e,1);      ....  "圆圆"的装置准备OK
  outport(0x322,0x02,1);      ....  "圆圆"切换到"初级"
  outport(0x321,0xff,1);      ....  8个LED全处于"开"状态
  }
```

☞ 请参照p.92的"致读者"。

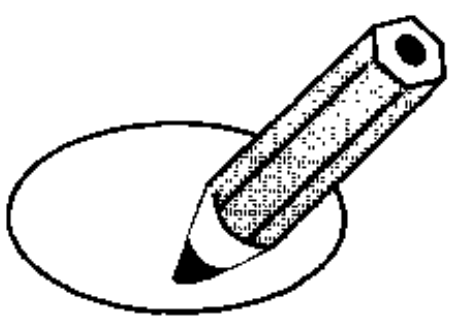


表 3 列出了 40 针插卡连接器的各信号。

表 3　40 针插卡连接器的信号

针　号	符　号	名　称	针　号	符　号	名　称
1	V_{CC}	+5V	2	V_{CC}	
3	ALE	ADRESS LATCH ENABLE	4	CS1	CHIP SEL 1
5	CS2	CHIP SEL 2	6	RSTO	RESET OUT
7	V_{IN}	VCC IN	8	INT	INTERRUPT
9	WR	WRITEENABLE	10	RD	READ ENABL
11	A17	ADDRESS 17	12	A16	ADDRESS 16
13	S1	STATE 1	14	S2	STATE 2
15	D7	DATA 7	16	D6	DATA 6
17	D5	DATA 5	18	D4	DATA 4
19	D3	DATA 3	20	D2	DATA 2
21	D1	DATA 1	22	D0	DATA 0
23	A15	ADDRESS 15	24	A14	ADDRESS 14
25	A13	ADDRESS 13	26	A12	ADDRESS 12
27	A11	ADDRESS 11	28	A10	ADDRESS 10
29	A9	ADDRESS 9	30	A8	ADDRESS 8
31	A7	ADDRESS 7	32	A6	ADDRESS 6
33	A5	ADDRESS 5	34	A4	ADDRESS 4
35	A3	ADDRESS 3	36	A2	ADDRESS 2
37	A1	ADDRESS 1	38	A0	ADDRESS 0
39	GND		40	GND	

① 即使计算机主开关 SW 处于断开状态，V_{CC}也常供电。

② V1N 为外部电源引线。对其施加 7～15V 电压可防止袖珍计算机内部的电池消耗。

③ S1，S2 随 CPU 动作模式，变化如下：



	S2	S1
MEMORY READ	1	0
MEMORY WRITE	1	1
IO　READ	0	0
IO　WRITE	0	1

40 针插卡连接器一边的 PPI 接口板，商品名为“宽宽 Z”，其全套组件及成品有售。

请向日本 YOUPIT 电子公司咨询(参见 p. 30)。

Ⅳ 接口电路

将来自袖珍计算机的控制信号连接到外部电路的过渡电路称为接口电路。典型的接口IC是美国Intel公司的名为PPI-82C55A的IC。

(1) PPI-82C55A

PPI是Programable Periperl Interfase(可编程外围接口)的字头缩写。其意义为可编程外部接口。另外,承担此PPI主要任务的IC是名为82C55A的IC。

82C55A,作为8位微机IC8080用的外部LSI,是美国Intel公司开发的制品。但是,关于今天的使用情况,8080中几乎不用,已被改良为8085及Z-80(Zilog公司)也可使用而得到广泛应用。82C55A为C-MOS型的LSI,用于低电流的场合。

82C55A中,有A,B,C三个8位并行I/O口,总计可进行24位输入输出。

24位输入输出,可用软件程序来决定每一位是输入还是输出。此即是所谓的可编程(programable)。被设置为输出的引脚,袖珍计算机的CPU可通过其输出"高"或"低"的电压信号,而被设置为输入的引脚,则可由其输入"高"或"低"的电压信号。

"高"电压信号指的是2V以上,程序将其视为"1"。而"低"电压信号指的是0.8V以下,程序将其视为"0"。图6是82C55的引脚配置图。

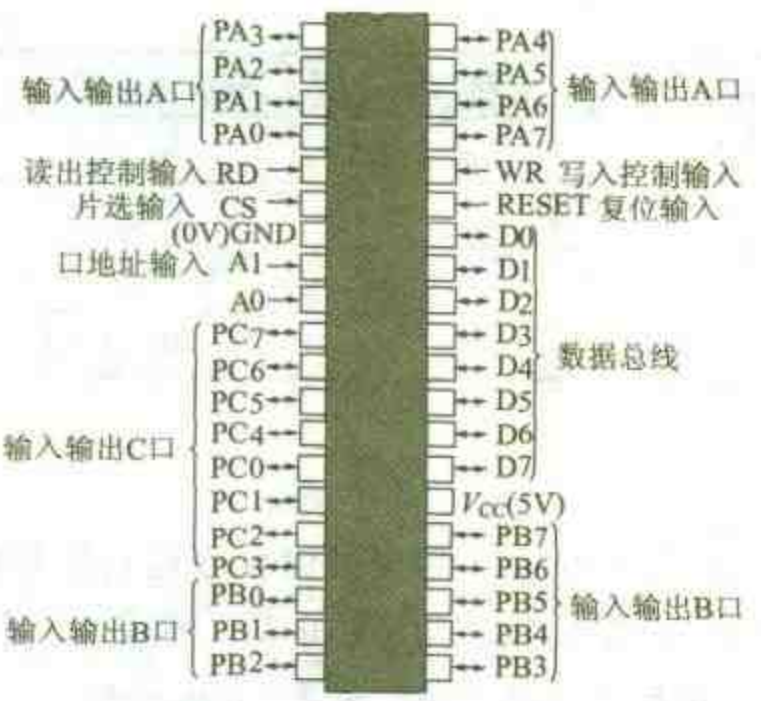




图6 82C55的引脚配置图

(2) 82C55A与8255A的差异

82C55A是8255A的C-MOS板。因而是具有C-MOS特点的低电流工作型

的 IC。对于像用电池工作的袖珍计算机那样的小电源装置，是非常合适的 IC。

表 4 给出了 82C55A 与 8255A 的差异。从数据上能明显地看出 I_{CC} 的不同。

表 4 82C55A 与 8255A 的差异



	8255A		82C55A	
	M_{IN}	M_{AX}	M_{IN}	M_{AX}
V_{IH} /V	+2.0	Vcc	+2.2	Vcc+0.5
V_{IL} /V	−0.5	+0.8	−0.5	+0.8
V_{OH} /V	+2.4		+2.4	
V_{OL} /V		+0.45		+0.45
I_{CC} /mA		120		5.0
I_{DAR} /mA	−1.0	−4.0	−1.0	−5.0

(3) 来自 30 针扩展引脚的口地址

口地址的设定，随计算机的种类而异。作为 30 针外部扩展引脚输出的信号，Z-1、VX 系列的袖珍计算机，以及 FX-870P，840P，860P，870P，890P 袖珍计算机具有相同的引出。即均按地址总线的 A0，A1，A2 与片选 CS3 的组合来决定。

“宽宽”C 系列及 mini“宽宽”的场合，由于只用一片 82C55A，其地址分配即如表 5 所示。

表 5

CS3	A2	A1	A0	地 址	口
L	L	L	L	00	A 口
L	L	L	H	01	B 口
L	L	H	L	02	C 口
L	L	H	H	03	控制字寄存器

(4) 82C55A 的模式设定

由于 82C55A 是所谓的“可编程外部 IC”，因而可由程序进行种种设定。这里的设定指的是模式设定，可选择模式 1、模式 2、模式 3 三种模式。模式 1、2，是按使用与外设间的数据传送及中断插入处理的“握手”方法进行。

模式 0 是最基本的使用方法。以下对此模式 0 进行说明。

(5) 作为模式 0 的 82C55A 的各口的输入输出

作为模式 0 的 82C55A 的各口的输入输出，按图 7 所示方法进行。

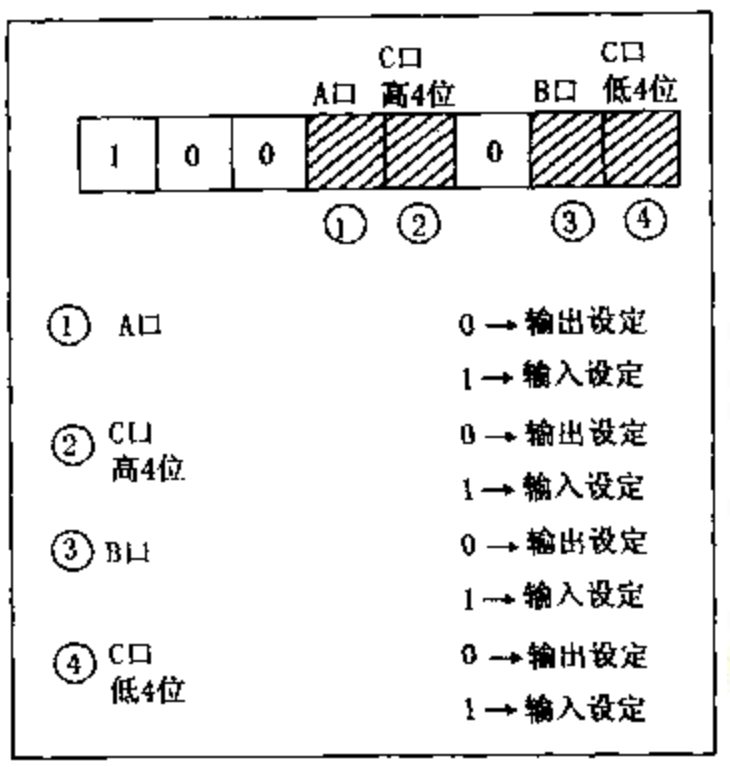




图7 模式0的各口的输入输出

由斜线所示的①、②、③、④四位的组合来决定各口的输入输出。图7①、②、③、④以外，按图所示进行“1”，“0”设置。此即模式0的使用方法。

图7的①，②，③，④输入的信号为0，1，1，1时，其设置即成：

A口：输出

B口：输入

C口：输入

则其控制字的数值码即为0x8B(二进制时为10001011)。

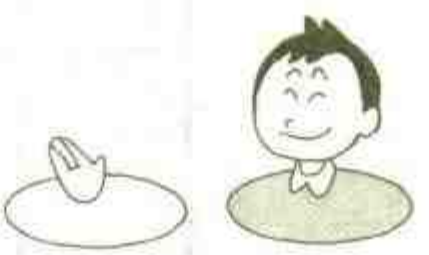
用于“圆圆”的控制字为0x90，它的设定为

A口：输入

B口：输出

C口：输出

从表7也可以看出其数值码为0x90。



下面，按照例题来练习一下各口的输入输出控制字的数值码都是多少。表6是练习题，表7是其答案。

表 6　控制数值码的练习

练　习	输入输出口				控制数值命令								
	A	C 高位	B	C 低位	二进制数								十六进制码
例题	出	出	出	出	1	0	0	0	0	0	0	0	80
练习1	出	出	出	入	1	0	0			0			
练习2	出	出	入	出	1	0	0			0			
练习3	出	出	入	入	1	0	0			0			
练习4	出	入	出	出	1	0	0			0			
练习5	出	入	出	入	1	0	0			0			
练习6	出	入	入	出	1	0	0			0			
练习7	出	入	入	入	1	0	0			0			
练习8	入	出	出	出	1	0	0			0			
练习9	入	出	出	入	1	0	0			0			
练习10	入	出	入	出	1	0	0			0			
练习11	入	出	入	入	1	0	0			0			
练习12	入	入	出	出	1	0	0			0			
练习13	入	入	出	入	1	0	0			0			
练习14	入	入	入	出	1	0	0			0			
练习15	入	入	入	入	1	0	0			0			



为使控制板“圆圆”工作，对爱称为“宽宽”的接口板的 82C55A 的 A、B、C 口设定如下：

A：输入

B：输出

C：输出

则按此设定的控制字如表 7 的网线部分所示，为十六进制数的“90”。

表 7　答　案

二进制数								十六进制码
1	0	0	0	0	0	0	0	80
1	0	0	0	0	0	0	1	81
1	0	0	0	0	0	1	0	82
1	0	0	0	0	0	1	1	83
1	0	0	0	1	0	0	0	88
1	0	0	0	1	0	0	1	89
1	0	0	0	1	0	1	0	8A
1	0	0	0	1	0	1	1	8B
1	0	0	1	0	0	0	0	90
1	0	0	1	0	0	0	1	91
1	0	0	1	0	0	1	0	92
1	0	0	1	0	0	1	1	93
1	0	0	1	1	0	0	0	98
1	0	0	1	1	0	0	1	99
1	0	0	1	1	0	1	0	9A
1	0	0	1	1	0	1	1	9B

(6) 袖珍计算机控制梗概

应用袖珍计算机进行控制的典型系统，如图 8 所示，是由“宽宽”系列或“宽宽”与控制电路“圆圆”构成。

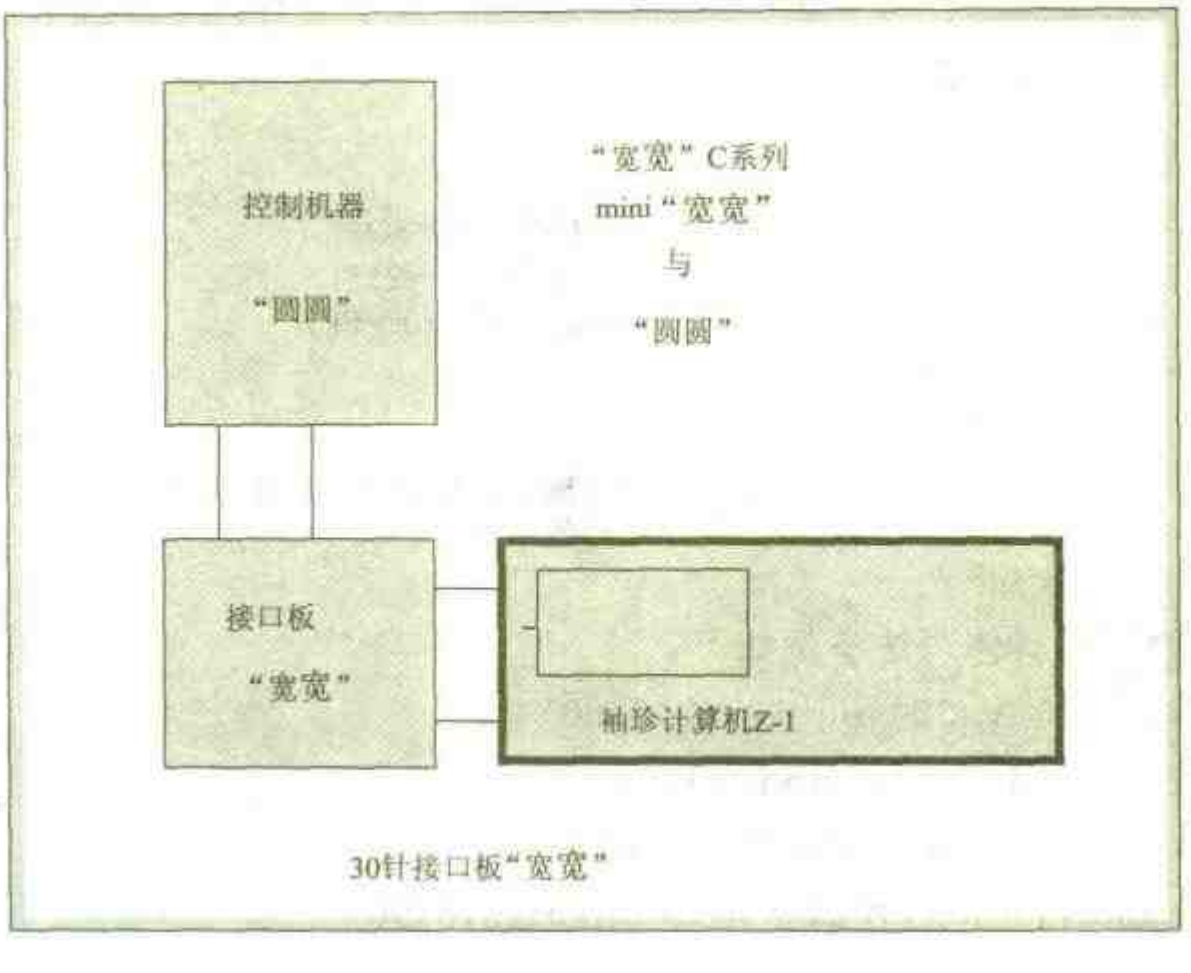


图 8



(a) 输出信号

由袖珍计算机的 CPU 向 82C55A 集成电路芯片发送信号，是将此信号以电压的高低输出，传送给 PPI 控制板，然后对外部控制机器进行控制。

(b) 输入信号

相反，另一个是得到外部控制机器的信号，将其传给 PPI 控制板，供袖珍计算机的 CPU 进行判断。

将 24 位口全部接到插头连接器，这样可以很自由地用扁平电缆将 CPU 周围的控制器移向外部。

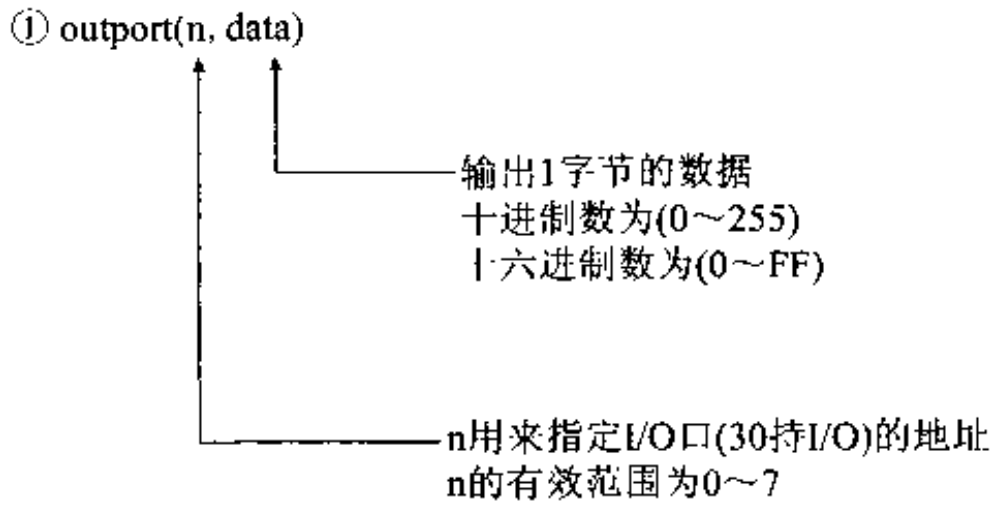
V 袖珍计算机的输出信号

如图 8 所示，可以从袖珍计算机向外部输出信号。

C 语言中，预先准备有输入输出库函数。所谓库函数，指的是使用袖珍计

算机的用户无需定义而在系统中已经定义有的函数。

所谓函数，以 BASIC 语言为例，其与“命令”的意思一致。但是不说成命令。命令只能决定一件事情。而函数在()中给予各种不同的内容则能够做许许多多的工作。因此在 C 语言中，几乎没有所谓命令的概念。



用 C 语言的具体写法举例如下：

输出 0x80 到 A 口时，outport(0,0x80)

输出 0x20 到 B 口时，outport(1,0x20)

输出 0xff 到 C 口时，outport(2,0xff)

输出 0x90 到 CWR(控制字)时，outport(3,0x90)



C语言控制“圆圆”录相带

将“宽宽”和“圆圆”接到 CASIO 袖珍计算机上、帮助用户轻松学习 C 语言的录相带正在出售。

请向日本 YOUPIT 电子公司咨询(参见 p. 30)。





该录像带对控制的基础知识进行了具体的讲述，广泛收集有从 C 语言入门到应用程序等内容。配合教材一起使用，C 语言的控制以及 C 语言编程的学习效率定会大大提高。

(该录相带播放时间为 58 分 33 秒。)

Ⅵ 正确使用"圆圆"

"圆圆"是由来自CASIO袖珍计算机VX系列以及Z-1的30针引脚的控制器推动的LED闪烁轮盘的爱称。来自10针卡边的控制器也能推动，但此时必需称为"宽宽Z"的PPI板。

利用CASIO袖珍计算机和此"圆圆"，可以理解控制电路的基本组成，并用C语言程序来建立直观明了的控制装置。

通过LED闪烁的种种状态及其速度的控制，以及演奏音阶的程序等，会在不知不觉中掌握C语言程序的学习。

此"圆圆"是为了各位用户能够自如地编制程序而刻意制作的。如果能理解A、B、C口的分配，编制程序一定会变成一件快乐的事情。好像很复杂的程序，实际上不过是基本逻辑的重叠堆积。基本而且扎实的编程是很重要的。



Z-1以外的袖珍计算机的控制

VX系列以及FX 840P、FX-860P、FX-870P等袖珍计算机均能用BASIC语言实行控制。但要在程序的开头先设置DEFSEG－&H1000后进行。

本书中使用的BASIC语言中的OUTPORT命令，是Z-1及FX-890中用的，应将此OUTPORT命令改为POKE语句。

INPORT命令也应改为PEEK语句。

VX系列 FX-840以下系列	Z-1袖珍计算机 FX-890
10 DEFSEG=&H1000	
20 POKE 3,&H90 …	…OUTPORT(3&,H90)
30 POKE 2,&H3E …	…OUTPORT(2,&H3E)
100A=PEEK(0) …	…A=INPORT(0)
	但是，即使是Z-1及FX-890袖珍计算机，POKE、PEEK命令也能使用。

(1)"圆圆"的构成

图9为"圆圆"的电路图。

图10为"圆圆"的实物结构图。

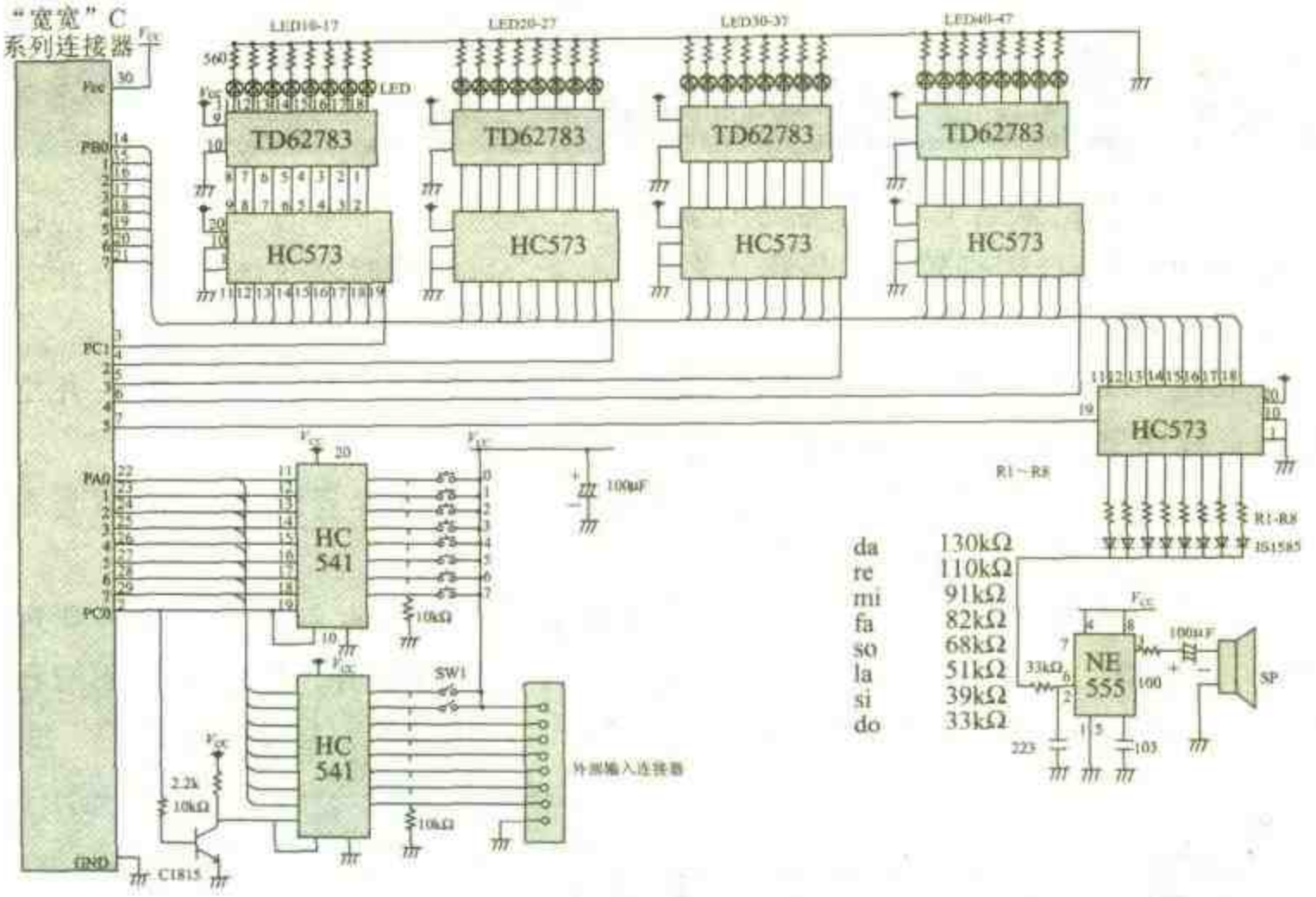


图 9　"圆圆"的电路图

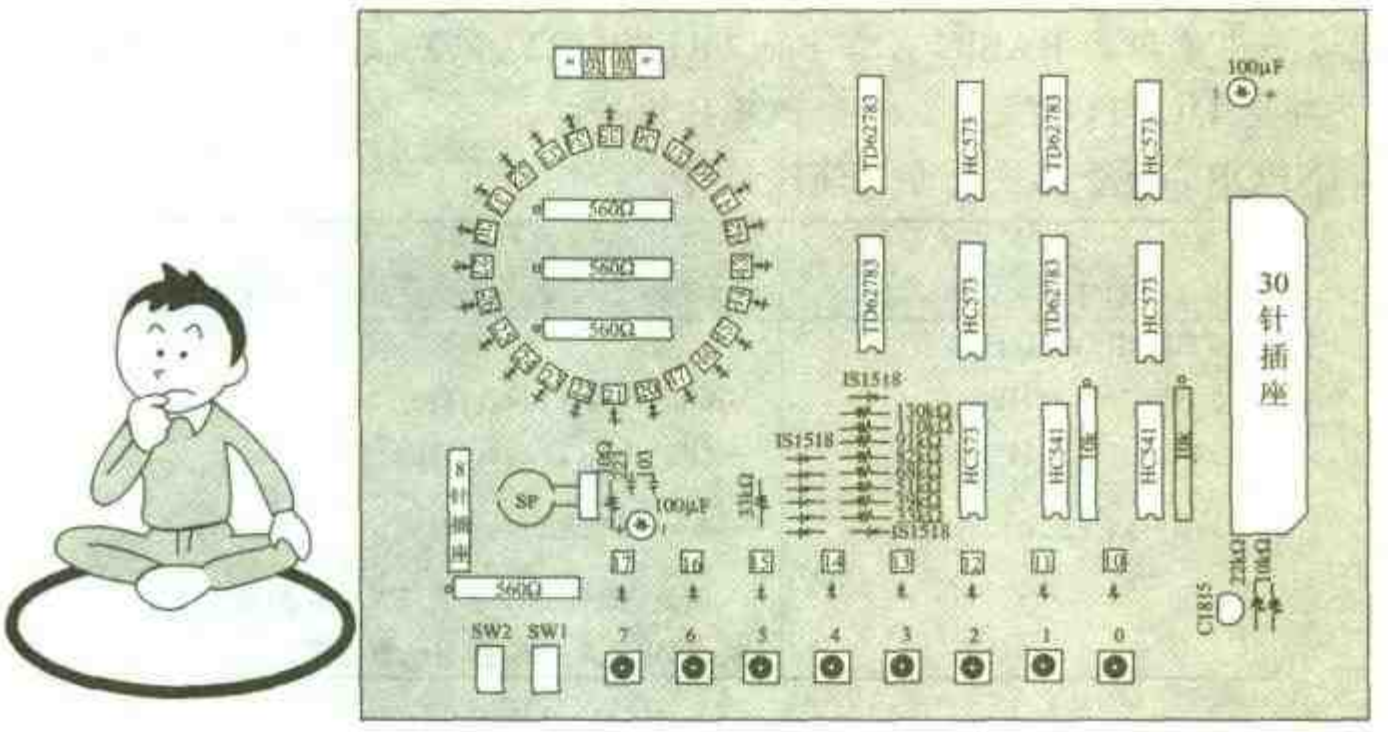


图 10　"圆圆"实况图

"圆圆"是将 24 个 LED 排成圆形，让其闪烁。LED 顺序闪烁呈现转动的圆圈时，恰如看轮盘赌的感觉。而且也能同时发出音阶。根据编程可做许多有趣的事情，是能够在娱乐的同时进行程序学习的好帮手。

为使LED闪烁，使用了4个74HC573锁存IC和4个8晶体管阵列来进行。为使LED闪烁时产生音阶，使用锁存IC和定时器IC555，以产生1个八度音阶的"do re mi"。

另外，作为输入用，还准备了2组8位输入，即8个排成1列的按键开关组和2个拨钮开关(2位)以及6个(6位)可选择安装的共计8位的输入组。

为平稳进行输入，还安装有2个74LS541缓冲器IC。

那么，和1排8个按键开关对应安装的8个LED是干什么用的呢？这是为实行"圆圆"的程序控制前进行"热身运动"而设计的。将其称为"圆圆"初等生也行。用C语言最初进行编程时，即可从此开始。

来自"宽宽"的3个口，与"圆圆"的关系如下：

A口：输入(按键开关等)
B口：输出(LED，奏乐)　} 控制字为0x90(详见表7)。
C口：输出(控制器)



接上mini"宽宽"或C系列"宽宽"的接口板，即可一步一步地开始编程。

(2)"圆圆"的口规格

(a) A口为输入口

A口为输入口，根据C口的0位(PC0)信号是高是低可进行两组8位输入的切换(参见表8、表9)。

表8　PC0低时

7位	6位	5位	4位	3位	2位	1位	0位
按开关	按开关	按开关	按开关	按开关	按开关	按开关	按开关

表9　PC0高时

7位	6位	5位	4位	3位	2位	1位	0位
						拨钮开关	拨钮开关

可选择6位外部输入

(b) B口为输出口

控制A口及C口的信号(参见表10)。

(c) C口是控制器

C口是输出口，是对连接到B口的LED及音阶进行选择的控制器(参见表11)。

表10

1排8个LED

PC1高,用0x02控制

位	7	6	5	4	3	2	1	0
	LED	LED	LED	LED	LED	LED	LED	LED

PC2高,用0x04控制

位	7	6	5	4	3	2	1	0
	LED	LED	LED	LED	LED	LED	LED	LED

PC3高,用0x08控制

位	7	6	5	4	3	2	1	0
	LED	LED	LED	LED	LED	LED	LED	LED

PC4高,用0x10控制

位	7	6	5	4	3	2	1	0
	LED	LED	LED	LED	LED	LED	LED	LED

"圆圆"的24个LED

音阶

PC5高,用0x20控制

位	7	6	5	4	3	2	1	0
	do	si	la	so	fa	mi	re	do

表 11

十六进制码	7 位	6 位	5 位	4 位	3 位	2 位	1 位	0 位
	未使用	未使用	音阶切换 L:OFF H:ON ♪	8 个	8 个	8 个	1 排 8 个	A 口输入切换
				"圆圆"的24个LED				
0	0	0	0	0	0	0	0	0
01	0	0	0	0	0	0	0	1
02	0	0	0	0	0	0	1	0
04	0	0	0	0	0	1	0	0
08	0	0	0	0	1	0	0	0
10	0	0	0	1	0	0	0	0
20	0	0	1	0	0	0	0	0
3E	0	0	1	1	1	1	1	0

由 C 口的控制器来决定接于 A 口及 B 口的装置。即通过 C 口的 0～5 位的开、关,对接于 A 口的输入开关组和接于 B 口的 LED 组进行切换。

如表 11 所示,根据 C 口的位的开、关,可对接于 A、B 口的装置进行切换。

十六进制码	说　明
0x0	释放所有装置。但只有A口的按键开关装置接着
0x01	A口的2个拨钮开关和选择输入装置接着
0x02	1排8个LED装置接着……练习程序用
0x04	圆盘上LED的第20～27接着
0x08	圆盘上LED的第30～37接着
0x10	圆盘上LED的第40～47接着
	这些单独连接的少。多是3组8位一起使用，用0x1c等进行指定
0x20	连接音阶装置……发出乐声
0x3E	连接B口的所有装置



Ⅶ “圆圆”开关ON!

按图8(参见p.37)所示连接袖珍计算机、“宽宽”和“圆圆”。

打开袖珍计算机的电源，其内部电源即会经由“宽宽”传给“圆圆”。但是，为确实保护袖珍计算机的电源，并不由它给“宽宽”供电，而是将外部电源接到“宽宽”，“宽宽”的内部则将外部切换开关置于“外部”。这样，即使打开袖珍计算机电源，“圆圆”的LED也不会开始旋转。

首先来做一些进行C语言编程的基本事情，以“圆圆”板上的1排8个LED的使用为例，活动活动肩膀。

“圆圆”开关ON时的故障处理

袖珍计算机、“宽宽”和“圆圆”接在一起时，如果有音阶发生及LED亮，可输入以下程序使系统处于复位状态。

C语言的情况

```
main()
{
      outport(3,0x90);
      outport(2,0x3e);
}
```

BASIC语言的情况

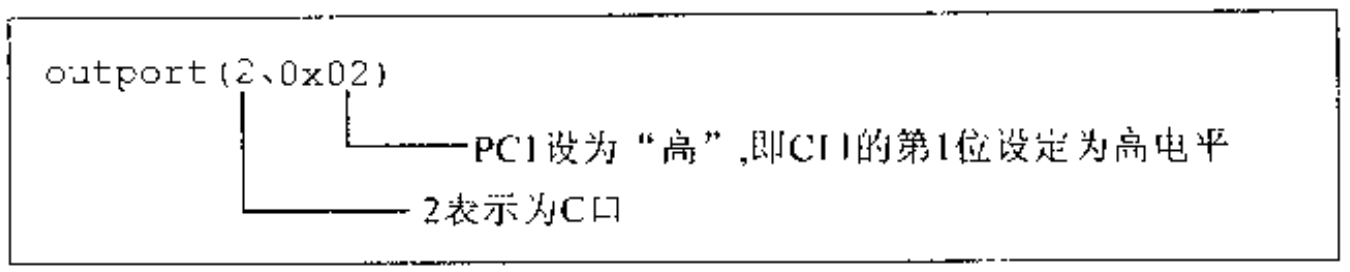
```
10  OUTPORT (3,&H90)
20  OUTPORT (2,&H3E)
```



(1) 8 个 LED 的亮、灭

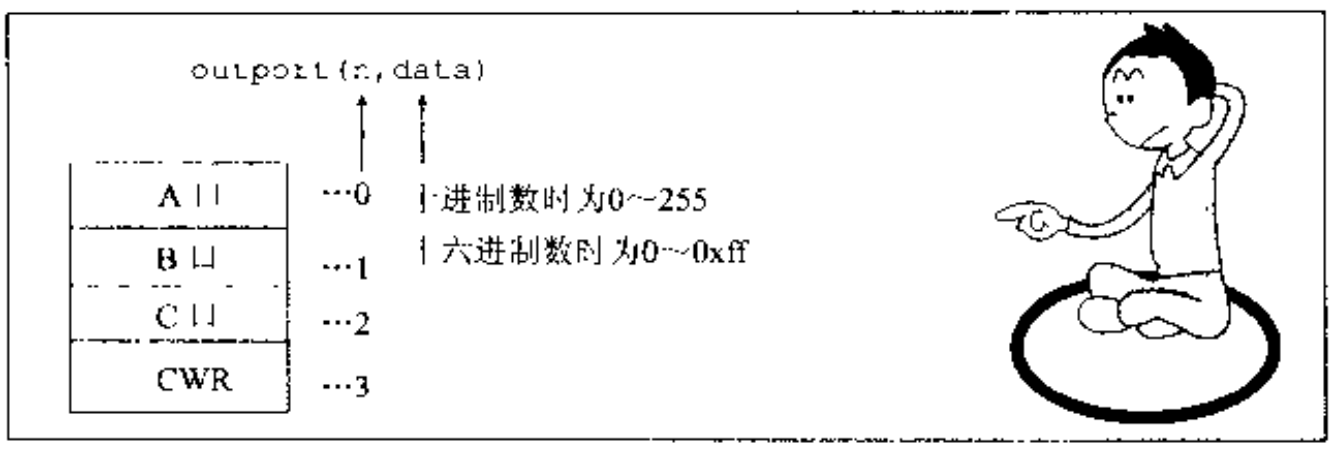
要让"圆圆"板上 1 排 8 个 LED 亮时，将 C 口的第 2 位(PC2)设定为"高"。

outport(　)函数，为通过接口电路从袖珍计算机向外部设备输出信号的函数。



练习题 1.1

编一段程序，使 8 个 LED 全亮，并经过定时器确定的一定时间后再全灭。



(CWR 为控制字寄存器，用以放置决定 A、B、C 各口为输入还是输出的编码(参照 P43 进行练习)。

●程序的编制与说明

```
/* M·W·R·K.C1 */                    .../*此中为注释*/
main(  )                            ...main(  )函数
{
  int k;                            ...表明变量 K 为整型变量
  clrscr( );                        ...清屏                    参见 p.36
  outport (3,0x90);                 ...A:输入,B:输出,C:输出(控制器)
  outport (2,0x3e);                 ...全部装置已准备好  参见 p.43
  outport (2,0x02);                 ...1 排 8 个 LED 复位  参见 p.43
  outport (1,0xff);                 ...1 排 8 个 LED 全亮
  for(k = 0;k <= 10000;k ++ );      ...延时
  outport (1,0);                    ...1 排 8 个 LED 全灭
}
```

/* */中的内容为注释。

/*M·W·R·K·C1*/程序，是用一个main()函数程序写的最基本的程序。将此基本程序作种种发展，C语言程序即得到拓宽。

例如，下面的text1.c程序中，是编写文字型指针以及程序的人将其编写成独立函数，以使程序更为简洁。

请参考程序右侧的说明来理解程序的内容。

练习题 1.2

使用文字型指针编一段程序，将与M·W·R·K·C1程序相同的结果在袖珍计算机的显示器上显示为：

如果LED亮……[8个LED亮]

如果LED灭……[8个LED灭]

另外，再编写timer()等函数，

●程序的编写与说明

```
/* text1.c */
main()
{
char *led8, *on, *off;          :表明为字符型指针变量，见注释*1(p.46)
led8 = "8个 LED";               :由于表明为字符型指针变量
on = "亮";                      :故需指定字符串数据
off = "灭";
clrscr();                       :清屏函数(袖珍计算机专用库函数)
start();                        :初始设定函数，称为 start()
outport(2,0x02);                :C口的第2位开((ON)(1排8个LED)
outport (1,0xff);               :输出0xff给B口，即8个全亮
printf("%s%s\n",led8,on);       :见注释*2(p.47)
timer(2000);                    :称为定时函数 timer()
outport (1.0x0);                :8个LED全灭
printf("%s%s\n",led8.off);      见注释*2(p.47)
}
/* start */
start()                         :调用初始设定用函数 start()
{                               :设定A为输入，B,C为输出
outport (3,0x90);               :设定控制字寄存器为0x90
outport (2,0x3e);               :电源(ON时，LED亮，音阶不响，处于复
                                 位状态。
```

```
}
/* timer *
timer(t)   ;调用定时函数 timer()
int t;     ;在 timer()函数全域内,变量 t 可有效地以 16 位整型数使用。
{
int k;     ;k 值从 0 开始,每次加 1 进行循环重复处理,直至其值到 t。
for (k=0;k<t;k++);
}
```

(a) Clrscr(　)函数

Clrscr(　)是消去袖珍计算机的屏幕显示并将光标回归原来位置的屏显控制函数。

(b) int 说明

int 是变量的类型说明。int 表示为整数型。

(c) Char 说明

Char 是整数型的文字说明。进行类型说明时,在变量的前标有“*”是表示该变量为指针变量。在做过类型说明后的程序内,变量前不再标“*”。



(d) for (　)语句

for(　)是循环控制型的重复处理的表述。

I 的值从 0 到 7 逐一变化。

(e) while(　)语句

```
while(1)
  {
  }
```

表示无限循环重复处理。

(f) 指针

(注释 * 1)

指针具有指示首地址序号的作用。说明某变量为指针变量时,只在类型说明时给变量的前面标注 *(星号),即表明该变量为指针变量。如

```
char *led8,*on ,*off;
```

等,即是说明变量 led8,on,off 均为字符型指针变量。

类型说明以外的地方标有 * +变量(标有 * 的变量),不表示为指针变量,而是指针指示的地址内容,有存放数据的含义。mawarul.c 的程序中没有这样

的记述，但如果在程序的最后有以下记述，可以证明以上的说明。

```
printf("%c\n", *on);
```

执行此语句，变量on最前头的地址"亮"字被显示。如果再有记述

```
printf("%c\n", *(on+1));
```

那么最前头加1号地址处的"了"字被显示。总之，类型说明以外的标有*的变量不是指针变量，而是指针指示的地址处的数据。

程序中，取掉*（星号）即成为指针变量。

```
printf("%s\n",led8);
```

的LED8是指针变量。因而会从LED8的首地址一直显示到"¥0"的编码。例如，以下程序语句及其相应的显示为：

```
printf("%s\n",led8);        :8コノLEDガ(8个LED)
printf("%s\n",led8+1);      :コノLEDガ(个LED)
printf("%s\n",led8+2);      :ノLEDガ(LED)
```

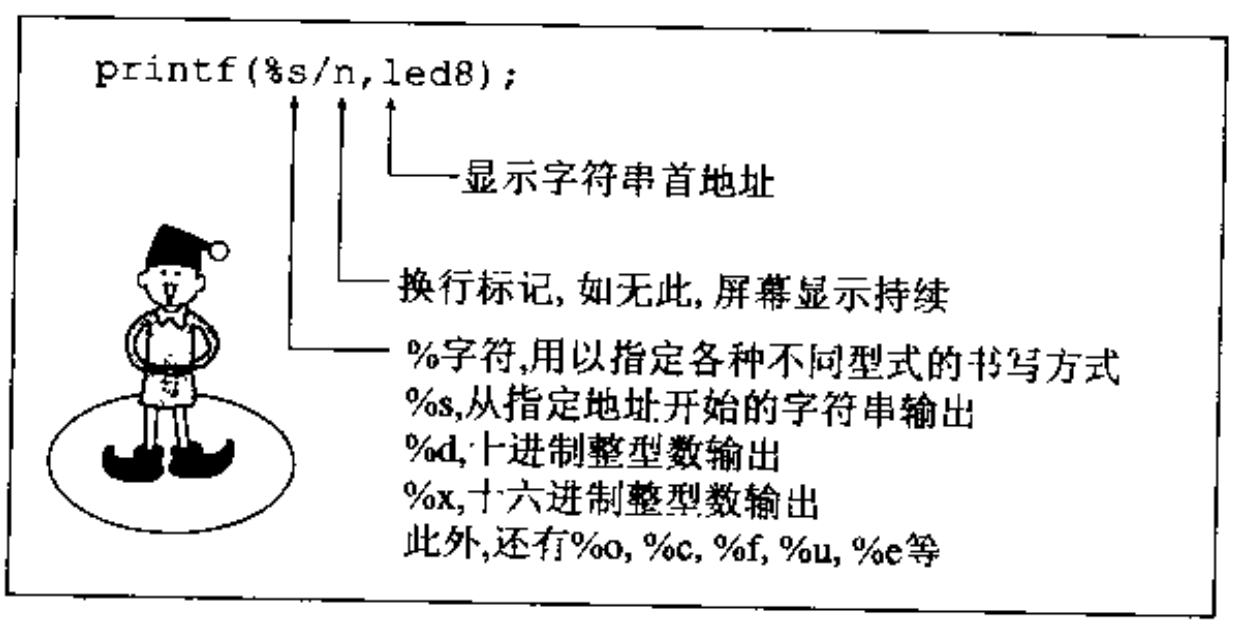
但是，即使看程序的每个地方，也看不到"\0"的记述。"\0"称为空码，即使程序用户不作记述，也会自动地添加上去。

(g) %字符

(注释*2)

printf("%s\n",led8);语句的说明

printf()是屏幕显示库函数。而库函数是指袖珍计算机内部已经备有的函数。



```
printf(%s%s/n,led8,off);
```

按此对应关系，还可追加内容，进行各种各样的表现

执行此程序，一排 8 个 LED 即亮。神珍计算机的屏幕上显示

8コノ　LEDガ　ツキマス(8 个 LED 亮)



延时一段时间后显示

8コノ　LEDガ　キエマシタ(8 个 LED 灭)

8 个 LED 熄灭。执行

```
outport (2.0x02);
```

C 口的第 1 位()N.B 口切换到 1 排 8 个 LED 状态时，执行

```
outport (1.0xff);
```

8 个 LED 全亮。

timer(2000)中的 2000 为自变量，程序处理到直至 timer(　)函数内的条件按此值满足。自变量改变时 LED 亮的时间会随之变化。t 值指定为 16 位整型数，其极限值为 32767。需要大于此值时，用 long t;代替 int t;这时指定的是 32 位整型数。



C 语言的格式



```
main(　)...main 是函数，所以必须要带(　)
{
    变量类型说明……进行所处理数据类型的说明
    int i;          ……说明变量 i 按整型数使用

    处理                进行处理
printf("ABC\n");     ;表示一条语句的终结。
            \n 是回车( ⏎ )的编码，省略时表示继续。
}…… main 函数结束标记
```

括号{　}表示 main 函数的有效范围。用{开始，用}结束。由于 C 语言中不使用引号，所以，这样的函数约束是必要的。

(2) 8个LED一个一个地亮灭变化

练习题1.3

编写一段程序，使1排8个LED从低位开始顺序亮灭。

●程序的编写与说明

学习程序/*M·W·R·K.C1*/，用一个main()函数完成程序的编写。

```
/*M·W·R·K.C2 */
main( )
{
    int i,k;
    clrscr( );
    outport (3,0x90);
    outport (2,0x3e);
    outport (2,0x02);
    while(1)                ……循环重复处理
    {
      for (i = 0x01;i<= 0x80;i<<= 1)……i的值逐位左移
      {
        outport (1,i);……随i值的变化，不同的LED亮
        for (k = 0;k<= 50;k++);……LED亮的时间短
      }
    }
}
```

初始设定，对1排8个LED的相关设定

此程序中无

outport (1,0);

因而1排8个LED不能一齐灭。

将/*M·W·R·K.2C*/进行整理，即知：

① LED不一齐亮；

② LED亮的形式不同；

③ LED亮的时间短；



④ 程序为无限循环。



(h) 左右移位算子

为让LED顺序亮灭闪烁,C语言程序中一般使用左右移位算子。

I≫1;变量I的数据右移1位。

I≫2;变量I的数据右移2位。

≫即为右移位算子。

左移位算子的符号为≪。

I≪1;变量I的数据左移1位。

C语言中备有大量的算子。一个一个地反复学习是很重要的。C语言程序的特点是不断地调用编制的函数,它是利用函数的程序。因而,在问题2的程序中,试编了尽可能多使用/ * mawaru 2.c * /这样函数的程序。

(i) while(1) { }的含义

```
/ * mawaru2.c * /
main ( )
{
    int i;
    clrscr ( );
    start ( );
    while(1)
    {
        outport (2,0x02);
        for (i=0x01;i <= ox80;i <<=1)
        {
          led(i);
          disp(i);
          timer(50);
        }
          printf( "\n" );
          outport (1,0x0);
    }
}
```

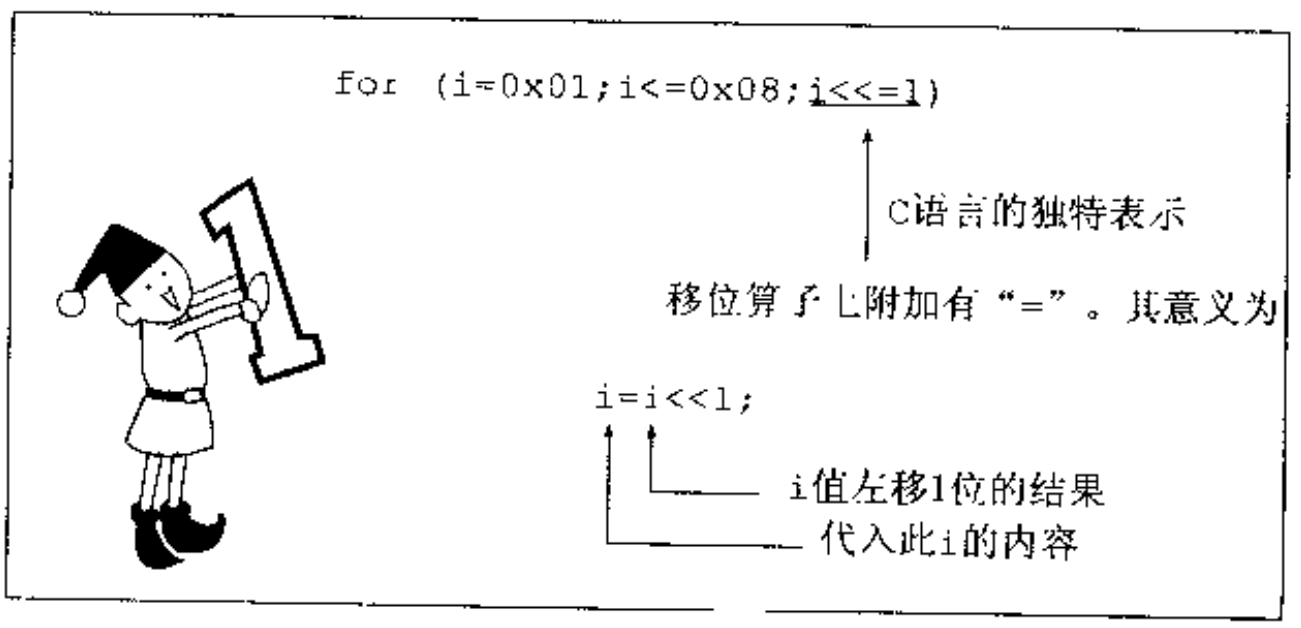
while(1)
{
此部分为无限循环
}

while(1)的()中为真值时,进行{}内的处理。while(1)的()内的值为1以上时常常为真值,因而会无限循环处理。

printf("\n")为袖珍计算机屏幕上8位1个字节的数据显示的换行。

```
/* start(  ) */                    :初始设定函数 ·····  (j)
start(  )
    {
    outport (3,0x90);
    outport (3,0x3E);
}
/* timer (  ) */                   :定时函数 ·····  (k)
timer (t)
int t;
{
    int k;
    for (k = 0; k<t;k++);
}
/* led (  ) */
led (x)                            :LED 亮灭函数 ·····  (l)
int x;
{
    outport (1,x);                 :输出至 B 口
}
/* disp(  ) */
disp (d)                           :袖珍计算机显示函数 ·····  (m)
{
    int i;
    printf("%2x",i);               :用十六进制指定 2 位
}
```

●程序的说明



(n) i≪=1 的含义

由于 for()是循环重复处理,上语句中的含义是 i 逐位左移,从 0x01(十六进制的 01)直至 0x80(十六进制的 80)为止。

(o) 编制函数

```
led(i);          :LED 点亮函数
disp(i);         :袖珍计算机显示函数
timer(50);       :定时函数
```

C 语言的程序中,编制特定的函数应使程序简练易读。这里编写了对 LED 点亮函数和袖珍计算机显示函数等,并冠以适当的函数名。

函数值与自变量密切相关。所谓自变量,指的是代入函数的()中的数值,即在函数之间进行交换的数据。

(p) 自变量

将自变量代入函数→习惯上称为赋值
从函数进行输出 →习惯上称为返回值

调用函数 timer(50);指的是将自变量 50 赋值给 timer()函数。再在 timer()函数中处理后返回到 main()函数内。

函数中按自变量的处理,分为三种类型:

函数 ① 不返回自变量的函数
函数 ② 以数值形式返回自变量的函数
函数 ③ 以地址或指针形式返回自变量的函数

以 LED 亮灭函数 LED(i)为例进行说明。

此函数是以数值形式返回自变量的函数。

```
  led(i);
      ↑
      └──────┐
/* led (i)*/  │  ledi(i)函数中自变量的变量名与被调用函数led(x)
led(x)←──────┘  的变量名不同或相同皆可。原因是一个函数内的变量
{                是独立的,传给函数的不是变量名(如int x;),而是变
                 量的取值
  outport (1,x);
}                    :输出至B口
```

●程序的执行

执行程序,1 排 8 个 LED 开始顺序亮灭闪烁。袖珍计算机的屏幕上显示出十六进制表示的内容为

1 2 4 8 10 20 40 80

由于是无限循环程序，不按断开键程序不会停止。

还有，进行以下的更改，可使程序变为亮了的 LED 不灭，8 个 LED 顺序变亮。

① `for (i = 0x01; i<= 0x80;i<<= 1)`

改为 0xff

②

```
i = |1;      :(i=i |1  取每位的 OR(或))
led(i);
disp(i);
```

③ disp()函数

```
printf("%2x",1);
```

设为3X
3表示是3位
大写字母X表示是十六进制

经过这样的改写，执行程序时，即会顺序变亮。袖珍计算机的屏幕上显示：

1 3 7 F 11 3F 7F FF



函数的比较

BASIC 语言	C语言
PRINT INPUT 称为命令语句 • • • MID$ RIGHT$ BASIC 语言中称这些为函数	此部分称为函数 ↓ printf("文字串\n",x,y); scanf("%d",&a); ()内称自变量

2 LED闪烁速度的调整与音阶

用C语言能够控制LED的闪烁，本节学习有关C语言的独特的函数及其结构。

所谓函数，即是可完成一件工作而整理成的程序。只要对实自变量及假自变量进行严格的处理，函数内处理的变量也可以与不同的程序连接使用。本章对有关函数的形式和程序的流程进行简单的说明 。

安装在袖珍计算机上的C语言，可单独运用函数。这样一来，特别有关控制的程序部分，只要将其整理成函数，使用就变得非常方便。

另外，在控制板的控制中，来自控制板的信号经袖珍计算机判别，其结果再去控制控制板上的LED及音阶。这也是控制的重要学习内容。

由控制板上的开关输入，体验LED闪烁的类型以及闪烁速度的变化的程序，可加深C语言的学习。

由于能发生音阶，将LED的闪烁与音阶配合在一起，一定能够编制出更有趣的程序。下工夫来编制有意思的程序吧！

I 函数集合体的编程

C语言就是函数的集合，下面学习其函数的有关内容。

练习题 2.1

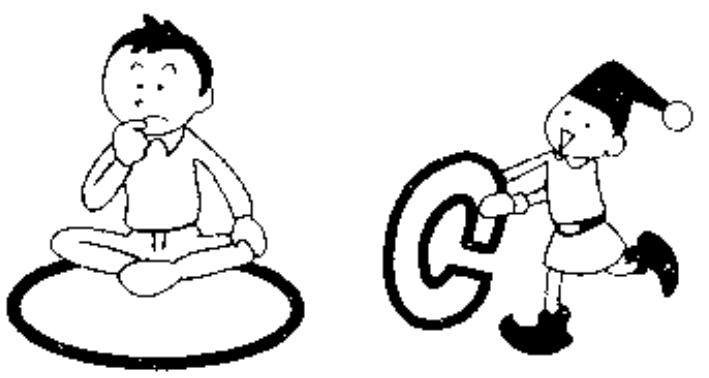
●编一段程序，使LED从下向上顺次闪烁后，再由上向下闪烁。并且在LED闪烁的同时，发出“doremi”音调。

用C语言试编的程序，即M·W·R·K.3C程序。

●程序的编写

```
  * M * W * R * K.3C *
main()
{
    int i.k;
    clrscr();
    outport (3.0x90);
    outport (2.0x3e);
    outport (2.0x22);                    ;音阶也能输出

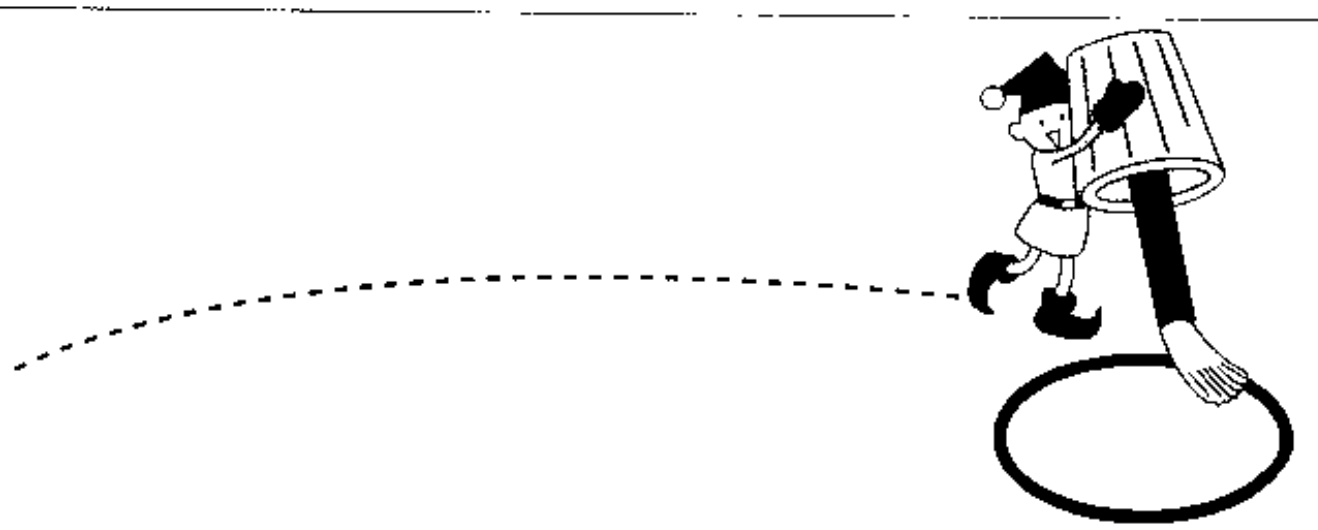
    while(1)
    {
      for(i = 0x01;i = 0x80;i = 1)       ;左移 1 位
      {
        outport(1.i);                    ;输出至 B 口
        for(k = 0;k = 50;k + + );        ;定时
      }
      for(i = 0x80;i = 0x01;i = 1)       ;右移 1 位
      {
        outport(1.i);                    ;输出至 B 口
        for(k = 0;k = 50;k + + );        ;定时
      }
      outport(1.0);                      B 口关(OFF)
    }
}
```



M·W·R·K.C'3程序是将定时部分当做独立的函数对待的。当然，执行的结果仍然相同。

```
/*M·W·R·K.C'3 */
main()
{
    int i;
    clrscr();
    outport (3,0x90);
    outport (2,0x3e);
    outport (2,0x22);

    while(1)
    {
        for(i = 0x01;i<= 0x80;i<<= 1)
        {
            outport(1,i);
            timer(50);              :将变量50连续传给定时函数
        }
        for(i = 0x80;i>= 0x01;i>>= 1)
        {
            outport(1,i);
          timer(50);
        }
    }
}
timer(t)                            :定时函数
int t;
{
    int k;
    for(k = 0;k<t;k++);
}
```



这里再从M·W·R·K.3C和M·W·R·K.3′C两个程序中，按独立函数进行分离，编制成的简洁程序即为M·W·R·K.C3。

```
/*M·W·R·K.C3 */
main()
{
    int i;
    start( );
    outport(2,0x22);
    while(1)
    {
        for(i=0x01;i<=0x80;i<<=1)
        {
            outport(1,i);
            timer(50);
        }
        for(i=0x80;i>=0x01;i>>=1)
        {
          outport(1,i);
          timer(50);
        }
    }
}
start()                    :初始设定函数
{
    clrscr( );
    outport (3,0x90)
    outport (2,0x3e)

}
timer(t)                   :定时函数
int t;
{
    int k;
    for(k=0;k<t;k++);
}
```



C语言可将函数集合、各种函数收集组成子构造，像搭积木那样，编制成程序。

Ⅱ 将函数部分分离为独立文件

练习题 2.2

●编一段程序，使 LED 从下向上顺次闪烁后，再由上向下闪烁。并且在 LED 闪烁的同时，发出“斗来米”音调。

但是，要将函数部分独立作成单独的文件，并用预处理功能语句 #include 进行指定。

(1) 预处理

C 语言处理系统中，具有对执行部分编译前进行预处理的功能。此即为

① #include 语句

② #defne 语句

#include 语句是将指定的文件内容原封不动地插入到有 #include 语句的地方。例如，想使用收在袖珍计算机的第 9 号文件中的文件时，作为预处理，在 main()函数前要写上

#include“f9”

此即表示的是分离为第 9 号文件的独立文件。



```
#include “f9”：记述指定的文件域名
/ * text2.c * /
main()
{
    int a,i,key;
    clrscr();
    start();
    printf(“按回车键 \ n”);
    outport (2,0x22);                    :指定连接到 B 口的装置
    while((key = getchar())! = EOF):袖珍计算机的键入条件，见注释 * 3 (p.61)
    {
    printf(“ % d % c \ n”,key,key);:输入键的十进制数表示和文字字符表
                                        示
        for(a = 0;a<= 7;a + + )          :使从 0 位到第 7 位变化
        {
            i = shiftl(a);               :左移位函数   注释 * 4(p.61)
            led(i);
```

```
            disp(i);
            timer(50);
        }
        printf("\ n");:袖珍计算机屏幕上 8 位 1 个字节数据显示的换行
    for (a = 0;a<= 7;a ++ )                :使从 0 位变到第 7 位
        {
            i = shiftr(a);                 :右移位函数
            led(i);
            disp(i);
            timer(50);
        }
        printf("\ n");                     :袖珍计算机屏幕上 8 位 1 个字节数
                                            据显示的换行
        outport (1,0x0);                   :停止向 B 口输出
    }
}
```

第 9 号的"f9"文件域中记述的即为下述程序。这些程序都是程序用户构造的函数。

```
/ * start() * /
start()                        :初始设定函数 ……(2)
{
    outport (3,0x90);
    outport (2,0x3E);
}
/ * timer () * /               :定时函数 ……(3)
timer(t)
int t;
{
    int k;
    for (k = 0;k<t;k ++ );
}
```



```
/* led() */                          :LED 亮灭函数  ……(4)
led(x)
int x;
{
    outport(1,x);
}
/* disp() */                         :袖珍计算机显示函数  ……(5)
disp(d)
int d;
{
    int i;
    printf("%2x",i);
}
/* shift1 */                         :左移位函数  ……(6)
shift1(x)
int x;
{
    int z;
        z = 1<<x;
        return(z);
}
/* shift(x) */                       :右移位函数  ……(7)
shiftr(x)
int x;
{
    int z;
    z = 0x80 >> x;
    return(z);
}
```



程序说明

(8) getchar()函数

(注释 * 3)

getchar()函数为库函数。所谓库函数,指的是予先准备好的函数。此函数为 1 字符输入函数,直至有键盘输入,它一直处于等待输入的状态。例如,输入 A,即是将 A 的 ASC 码"0x41"输给变量键。所谓 1 字符输入函数是说即使有 1 字符输入,此函数也不立即执行。被输入的字符编码,暂时存放在缓冲器中,直至回车键↵按下前,一直在进行存入。

按下回车键↵,此前的数据变得能输向计算机。↵也被看成"\n"。

程序执行时,getchar()函数对 1 字符输入,也可看作是字符串输入。

```
while((key=getchar( )) ! =EOF)
```

EOF 值为-1。在袖珍计算机中是 SHIFT 键+↵(个人计算机中的 EOF 是 CTRL+Z(control Z)。)。

key 的值只要不是-1,字符会不断存入缓冲存储器,↵信号来到,才进入下一步的处理。

↵信号的记号为"\n"。

程序执行中,从键盘输入某适当字符,按下↵键时,程序会按照输入键按下的次数,重复进行同样次数的处理。



(9) 实变元、假变元

(注释 * 4)

左移位函数

```
for(a=0;a<=7;a++)
 {
   i=shiftl(a);
    led(i);
    disp(i);
    timer(50);
 }
```

a: 实变元

a的值被传给x。

相互的变量名，同异皆可。原因是函数是各自独立的，没有因变量名不同就不能调用的道理

```
/* shiftl */
shiftl(x)
int x;
{
   int z;
   z = 1 << x;
   return(z);
}
```

x:假变元

运算处理的结果回传给变量z。变量z的值反映返回给变量i的值



如果这部分难理解时，可加上z并将i换写为z，即

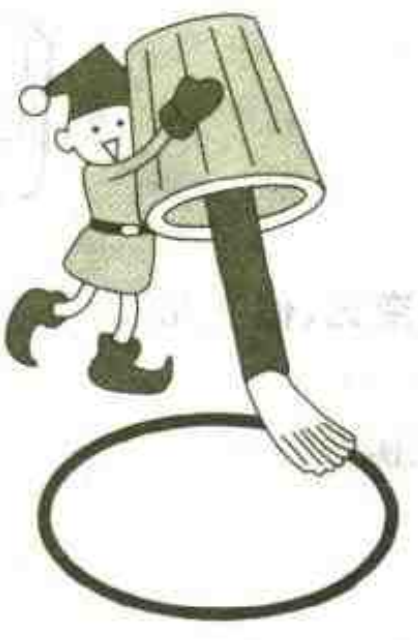
```
int a,i,key, z;
z=shiftl(a);
led(z);
disp(z);
```

加上z

将i换为z

而且，shiftl(　)函数还有下述写法：

```
/* shiftl() */
shiftl(x)
int x;
{
return( 1<< x);
}
```



●程序的执行

若执行程序，袖珍计算机的屏幕上立即显示：

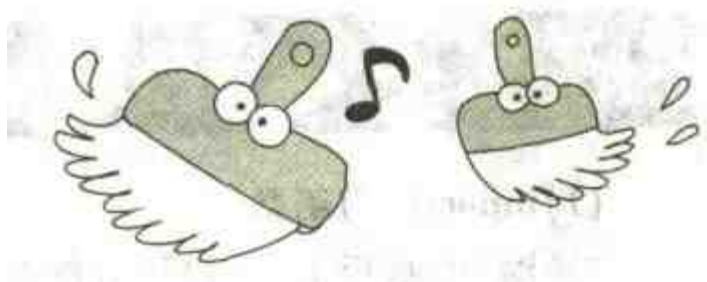
リターン キー ヲPUSH(按回车键)

照此去做，“圆圆”基板上安装的喇叭即会奏出“do remi fa so la si do—do si laso fa mi re do”的音阶。合在一起，袖珍计算机的屏幕上会显示：

1 2 4 8 10 20 40 80

80 40 20 10 8 4 2 1

发出音阶是由于有下述的设定。



```
outport (2,0x22);
```

2 … 十六进制数

0010 0010 … 二进制数

C口 ← 音阶装置ON ← 1排8个LED装置ON

1排8位的装置和音阶装置都接在B口，所以用led()函数，LED和音阶两者的输出可以兼顾。

```
/ * led () * /                          :LED亮灭函数
led(x)
int x;
{
     outport (1,x);
}                    ← X的值兼顾LED和音阶两者的输出。
```

Ⅲ “圆圆”的LED及音阶的停止

为停止LED及音阶，只要将C口的0～5位置为低电平，接于B口的装置即被释放。

使用下述语句，LED的回转闪烁及音阶即被停止。

```
outport (1,0);
```



Ⅳ　C 语言控制基础(输入篇)

(1)inport(　)函数

"转圈"的基板上,A 口被分配为输入口。A 口有 2 组输入,可用 C 口的第 0 位进行切换。

现在考虑,将 8 个按键开关逐一按下,让 1 排 8 个 LED 亮且发出音阶的情况。

必须将外部信号送给袖珍计算机的 CPU。此函数即是 inport(n)函数。该函数为输入输出库函数

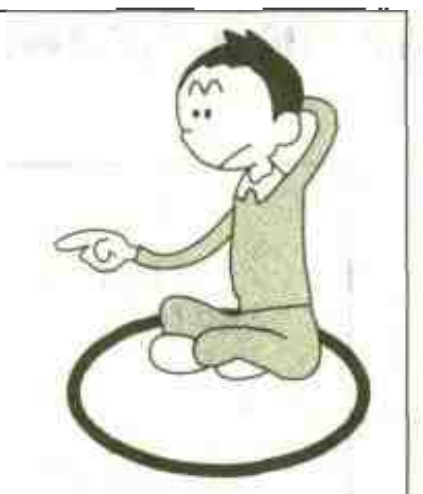
```
a=inport(n);
```

它从 n 指定的 I/0 口读入 1 个字节的数据。具体情况为

```
a = inport( 0 );
```

读入A口的内容
若为B口则为1
若为C口则为2
其有效范围为0～7

将口读入的 1 字节数据(0～255或0～FF)代入变量a



这样,通过 inport(　)函数,从外部读入的 1 字节数据即送给变量。输入了数据的变量由袖珍计算机处理。

练习题 2.3

●编一段程序,在 8 个按键开关按下时,让 1 排 8 个 LED 亮,并同时奏响音阶。并且在 8 个按键开关按下时,将数据显示在袖珍计算机的屏幕上。

使用 inport(n)函数,来进行"转圈"与 Z-1 间的"投接球练习"。inport(　)函数为输入控制函数,袖珍计算机 Z-1 有读入口数据的功能。设

```
a = inport(0);
```

则在 0 口即 A 口读取的值被代入变量 a。若有语句



```
outport(1,a);
```

则按照 inport()函数读取的 a 值，若装置连接有 LED 及喇叭，则 LED 变亮、喇叭发出音阶。

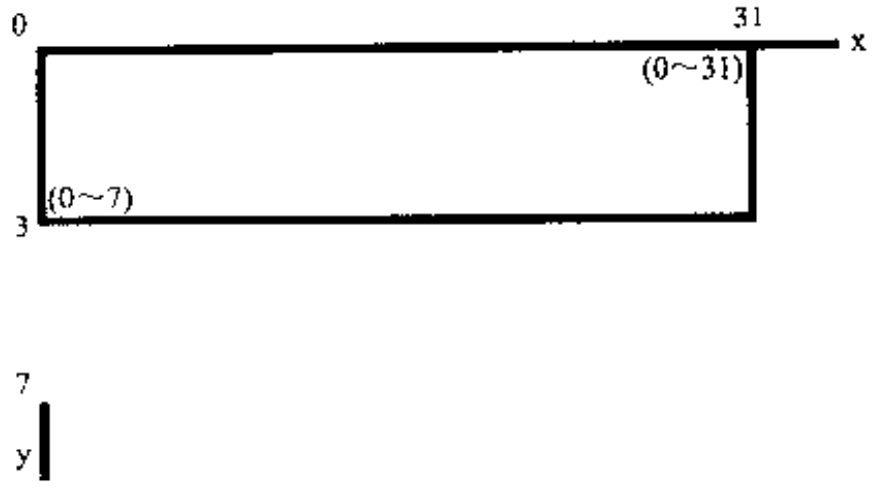
```
/* M·W·R·K.4C */
main()
{
    int a;
    start ();
    outport (2,0x22);

    while(1)
    {
        gotoxy(1,1);printf("PUSH SW 请按某 -开关");
        a = inport(0);
        gotoxy(1,2);printf("十六进制 data 为 0x%02x",a);

        outport(1,a);
        }
    }
start()
{
    clrscr();
    outport (3,0x90);
    outport (2,0x3e);
}
```

(a) gotoxy()函数

gotoxy(x,y)函数为屏幕控制函数，是用于控制袖珍计算机屏幕显示位置的函数。



练习题 2.4

●编一段程序，在按下按键开关时，由于有音阶发生，与其配合，让袖珍计算机的屏幕上显示"do、re、mi"音阶。（请将程序以编制成函数的形式，做成"f9"文件后附形。）

●程序的编写

```
/ * M • W • R • K.5C * /
main()
{
    int a,i,d1[8];
    char * dat1;
    dat1 = "01020408102040 80";
    start();
    outport (2,0x22);

    for(i = 0;i<7;i ++ ,dat1 += 2)
        sscanf(dat1," % 02x",&d1[i]);

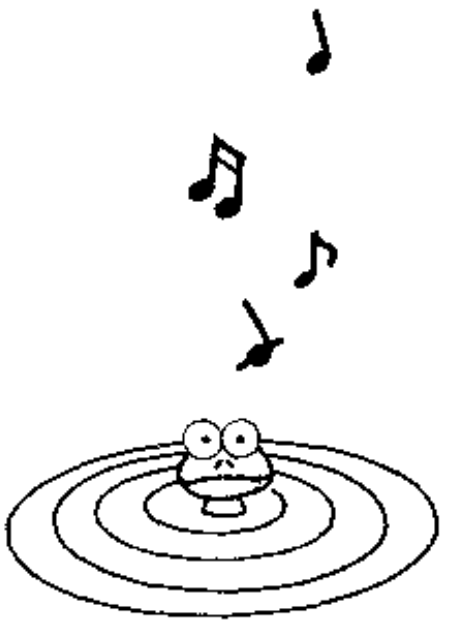
      while(1)
      {
          for(i = 0;i<= 7;i ++ )
          {
            outport (1,d1[i]);
            timer(200);
            outport(1,0);
            timer(300);
          }
```

简单的说明

"do re mi fa so la si do"
是8个数据。
d1[8]是d1矩阵说明，
字符型指针说明
音符数据
dat1=dat1+1

d1[1]=0x01　:do
d1[2]=0x02　:re
d1[3]=0x04　:mi
d1[4]=0x08　:fa
d1[5]=0x10　:so
d1[6]=0x20　:la
d1[7]=0x40　:si
d1[8]=0x80　:do

```
for(a = 0x80;a >= 0x01;a >>= 1)
            {
              timer(200);
              outport(1,a);
            }
            timer(200);
            outport(1,0);
            timer(500);
          }
}
start()
{
     clrscr();
     outport (3,0x90);
     outport (2,0x3e);
}
timer(t)
int t;
{
     int k;
     for(k = 0;k<t;k++);
}
```



●**程序的说明**

(b) 矩阵说明

```
int a,i,dl[8];                    dl[8]是名为 dl 的矩阵说明,准备有 8 个箱
                                  子放"do re mi fa so la si do"8 个数据。
char *datl;                       字符型指针说明,datl 的 * 为指针标记。

datl = "0102040810204080";        指针表示被存储变量的字头。
                                    01:"do",02:"re",04:"mi"
                                    08:"fa",10:"so",20:"la"
                                    40:"si",80:"do"十六进制数据
```

(c) scanf(　)函数与 sscanf(　)函数

dat1++2 是 dat1=dat1+2 的意思，是 C 语言独特的表达方法。

```
for(i=0;i<=7;i++,dat1 +=2)
    sscanf(dat1,"%02x",&d1[i]);
```

scanf(　)函数是由键盘输入，

sscanf(　)函数是由缓冲器输入。

sscanf(　)函数中的 dat1 的数据要两个字一组代入 &d1[i] 中，即，将 dat1 的数据两两分开，代入 d1 准备好的 8 个箱子中。



练习题 2.5

●编一段程序，使用 gotoxy(　)函数在屏幕上显示：

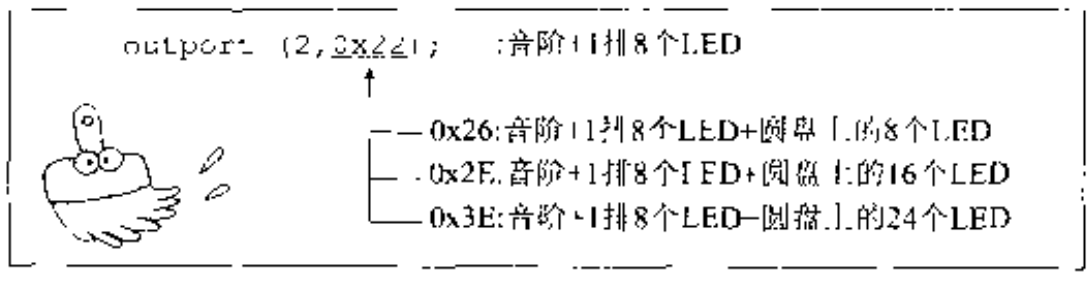
8コノPUSH SWノドレカヨオシテクダサイ(请按 8 个 PUSH SW 的某 1 只)

ソノトキノ16シンdata，%02xデス(此时的十六进制 data 为%02x)

```
#include "f9"
/ * text3.c * /
main()
{
    int a;
    clrscr();
    start();
    outport (2,0x22);              ;音阶和 1 排 8 个 LED
    while(1)
    {
      gotoxy(1,0);printf           ("请按 8 个 PUSH SW 的某 1 只 \ n");
      gotoxy(1,1);printf           ("此时的十六进制 data 为%02x",a);
      led(pushsw());
    }
}
```

●程序的说明

"F9"的文件中，要有新函数：push sw()函数的添加程序语句。



输出给 C 口的数据进行上述变更后，会连接上与之相应的装置。

mawaru1.c 程序中，修改 outport(2,0x22)的数据部分，即可对各种装置的连接进行确认。请按此去做。

(d) led(pushsw())函数

led()函数中，pushsw()函数变成了自变量。即用 pushsw()函数读入 A 口的数据，再将此数据作为自变量传给 led()函数。

led()函数是输出给 B 口的函数，因而会输出 LED 和音阶两方面。

"f9"文件的最后补充的程序语句为

```
/* pushsw() *                    :按开关函数
pushsw()
{
return(inport (0));          :读入 A 口的值
}
```

构造 pushsw()函数。按下 A 口上的按键开关时，其值用 inport()函数读入。可见，pushsw()函数是按下按键开关时读取 A 口值的函数。

```
/* start() */
start()                      :初始设定函数
{
    outport (3,0x90);
    outport (2,0x3E);
}
/* timer () */                :定时函数
timer(t)
int t;
{
    int k;
    for (k = 0;k<=t;k++);
```

```
}
/* led() */                    :LED亮灭函数
led(x)
int x;
{
    outport(1,x);
}
/* disp() */                   :袖珍计算机显示函数
disp(d)
int d;
{
    int i;
    printf("%2x",i);
}
/* shiftl */                   :左移位函数
shiftl(x)
int x;
{
    int z;
        z = 1≪x;
        return(z);
}
/* shift(x) */                 :右移位函数
shiftr(x)
int x;
{
    int z;
    z = 0x80 ≫ x;
    return(z);
}
```



(e) sound()函数

编写 sound()函数。此函数为按下按键开关后，显示“do”、“re”、“mi”并奏

响音阶的函数。此函数中，使用 C 语言的 switch～case 语句。

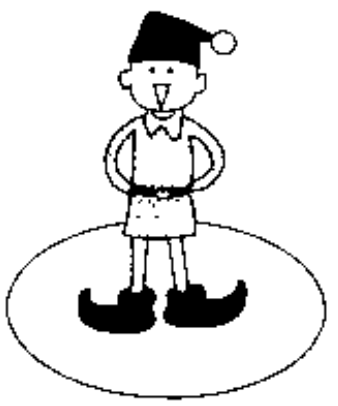
(f) switch～case 语句

switch～case 语句用于多分支的场合。

switch 语句是将无()的式子的值拿来和全部 case 进行比较。每种情况都用整数或文字常数标以标记。比较结果一致时，控制移向相应的 case 语句。所谓 default 语句，是用于对照各种 case 语句，不一致时进行处理。

要从 switch～case 语句中移出时，使用 break 语句。

```
/* sound() */
sound(i)
int i;
{
    int j;
    switch(i){
            case 0x01:printf("\n其音阶为"do"\n");
            break;
            case 0x02:printf("\n其音阶为"re"\n");
            break;
            case 0x04:printf("\n其音阶为"mi"\n");
            break;
            case 0x08:printf("\n其音阶为"fa"\n");
            break;
            case 0x10:printf("\n其音阶为"so"\n");
            break;
            case 0x20:printf("\n其音阶为"la"\n");
            break;
            case 0x40:printf("\n其音阶为"si"\n");
            break;
            case 0x80:printf("\n其音阶为"do"\n");
            break;
            defalt:i = 0;
        }
        outport(1,i,);
}
```



V　音阶变化的同时让 LED 旋转

练习题 2.6

●编制音阶变化的同时让 LED 移动的程序。

将音阶数据当做矩阵来处理。

```
#include "f9"
/* text4.c */
main()
{
    int a,i,d1[8];                    :d1[8]为一维矩阵说明
    char *dat1;
    dat1 = "01020408102040 80";       :音阶数据
    start();
    outport (2,0x22);
    for(i = 0;i<= 7;i ++ ,dat1 += 2)
    sscanf(dat1," %2X",&d1[i]);
    while(1)
    {
        for (i = 0;i<= 7;i ++ )
        {
            led(d1[i]);
            disp(i);
            timer(200);
            outport (1,0);
            timer(500);
        }
        printf(" \ n");
        for (a = 0;a<= 7;a ++ )
        {
            i = shiftr(a);            :原样利用右移位函数
            timer(20);
            led(i);
        }
    }
}
```



●程序的说明

(a) 一维矩阵

dl[8]是 C 语言的一维矩阵，使用的不是()而是[]。

for()是表示循环控制型的重复处理，i 值从 0 到 7 逐渐变化。

dat1 += 2 的含义是 dat1 = dat1 + 2，是 C 语言中特定的算子。

sscanf()，如果说 scanf()是指键盘输入，则 sscanf()为由缓冲器进行数据的输入输出。

& dl[i]的 & 称为地址算子，用于显示存储数据的存储器的地址值。

(b) "*"与"&"的区别

"*"表示指针，用于指示存有数据的存储器的头地址。

"&"则为存有数据的存储器的地址。

练习题 2.7

●通过种种方法已经明白了音阶的产生过程，试着来演奏一支曲子。试用单纯数据来演奏贝多芬的"欢乐颂"。

```
#include "f9"
/* text5.c */
main()
{
    int i,dl[65],key;
    char *dat1;
    dat1 = "0404081010080402010102040402020404081010080402010102
          0402010102020401020408040102040804020120010404081010
          0804020101020402010l";
  clrscr();
  start();
  outport (2,0x22);
  for(i = 0;i <= 61;i++,dat1 += 2);
  sscanf (dat1,"%2X",&dl[i]);
  printf("hit anykey \ n");
  while(key = getch() ! = EOF)
  {
       clrscr();
```

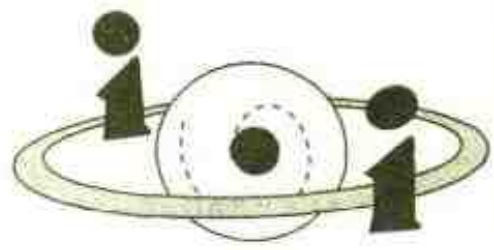
```
        printf("欢乐颂 \ n");
        for (i = 0;i<= 61;i ++ )
        {
            led(d1[i]);
            timer(50);
            outport (1,0);
            timer(50);
        }
    }
```



Ⅵ　C 语言编程技巧

练习题 2.8

●考虑下述问题的编程：让圆形排列的 24 个 LED 顺序闪烁，方向按顺时针，速度按 timer() 函数 120 的快慢，并同时反复演奏"do re mi fa"音阶。



```
/ * M • W • R • K.C2 * /   以此程序为参考
main()
{
    int i,k;
    clrscr();

    outport (3,0x90);
    outport (2,0x3e);      初始设定
    outport (2,0x3c);      有关 24 个圆形排列的 LED 的初始设定
    while(1)                                    ……循环重复处理
    {
      for(i = 0x01;i<= 0x80;i<<= 1)             …将 i 值逐位左移
      {
        outport(1,i);                           …LED 的亮灭随 i 变化而异
        for(k = 0;k<= 50;k ++ );                …LED 亮的时间短
      }
    }
}
```

于中有"圆圆"录像带的，请参考录像教材训练来考虑程序的编制。

```
  * M * W * R * K.6c * /
main()
{
    int i,p,p1;          i;用于 LED 及音阶 一个一个移动的变量
    clrscr();            p;为使 24 个 LED 3 个 一组接力闪烁的变量
    start();             p1;使音阶与 LED 时序吻合的变量
    while(1)
    {
        for(p = 0x04;p <= 0x10;p <<= 1)   3 组 24 个 LED 每 1 的移动
        {
          p1 = (p|0x20);              音阶与 LED 的闪烁同时进行
          outport (2,p1);
          for(i = 0x01;i <= 0x80;i <<= 1)      8 个 LED 顺序闪烁
          {
            outport(1,i);
            timer(120);
          }
          outport(1,0);
        }
    }
}
/ * start() * /
start()
{
    outport (3,0x90)
    outport (2,0x3e);
}
/ * timer() * /
timer(t)
int t;
{
    int k;
  for (k = 0;k<t;k + + );
}
```

●程序的说明

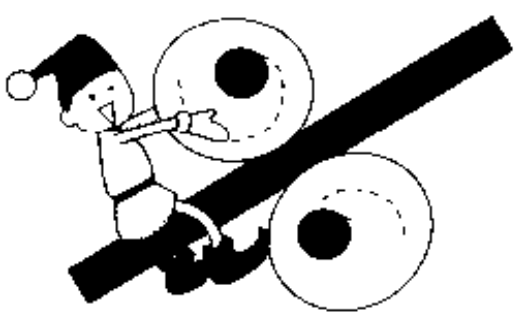
(a) C 语言的 OR 算子

```
for(p = 0x04;p  = 0x10;p = 1)
```

1 位 1 位移动，圆形排列的 LED 3 组 8 位的装置序号即为 0x04、0x08、0x10。

```
pl = (p : 0x20);
```

将音阶装置与圆形排列的 LED 的每一装置按 OR 求和。



```
pl = 0x04 OR 0X20.....0x24
pl = 0x08 OR 0X20.....0x28
pl = 0x10 OR 0X20.....0x30
```

pl 值的这种变化即是 C 语言中 OR 算子 : 作用的结果。

练习题 2.9

●圆形排列的 LED 闪烁和音阶发生的速度随定时器延时量而变化。作一段程序，从键盘输入定时数值，来观察速度的变化。



```
/ * text6.c *
main()
{
    int i,p,pl,t,key;
    clrscr();
    start();
    printf("按回车键 push \ n");
    while(key = getchar() !  = EOF)
    {
    printf("定时器设定 1000 以下数值 \ n");scanf(" % d",&t);
        for(p = 0x04;p = 0x10;p  = 1)
        {
            pl  = (p : 0x20);
            outport (2,pl);
            for (i = 0x01;i  = 0x80;i = 1)
        ;
            led(i);
```

```
            disp(i);
            timer(t);
        }
        printf(" \ n");
        outport (1,0);
        }
    }

/ * start() * /                    :初始设定函数
start()
{
    outport (3,0x90);
    outport (2,0x00);
}
/ * timer () * /                   :定时函数
timer(t)
int t;
{
    int k;
    for (k = 0;k < t;k ++ );
}
/ * led () * /                     :LED 闪烁函数
led(x)
int x;
{
    outport (1,x);
}
/ * disp() * /                     :袖珍计算机显示函数
disp(d)
int d;
{
    int i;
    printf(" % 2x",i);
}
```





Z-1机中由40针插卡连接器进行控制

作为来自40针插卡连接器的PPI板，有"宽宽"Z-30接口板。

此时，由于其与杜邦公司制造的30针扩展连接器的地址不同，即使按本教科书的程序原样进行执行也不动作。请按以下所示改写部分程序后再执行(参见 p. 36)。

(1) C语言的情况下

增加 →

```
outport(0x90,0x300,2);
outport(0x92,0x408,2)
```

← 将此部分放在程序的前头

```
outport(3,0x90);→ outport(323,0x90);
outport(2,0x3e);→ outport(322,0x3e);
outport(2,0x22);→ outport(322,0x22);
```

作这样的更改

```
inport(0);→ inport(320); 改为后者
```

来自杜邦制30针扩展连接器的信号时的0～3地址应作如下变更：

0 → 320
1 → 321
2 → 322
3 → 323

(2) BASIC语言的情况下

初始设定时为：

```
OUT &H90,&H300,2
OUT &H92,&H408,2
```

因此请将OUT &H3,&H90改为OUT &H323,&H90。
余类同。

0 → 320
1 → 321
2 → 322
3 → 323

详细情况请参见本书P.30"40针插卡连接器"。

3 LED闪烁的缓慢停止

控制板上，除掉8个按键开关外，还装有两个拨钮开关。此前，讲述了LED的各种闪烁以及伴随闪烁的音阶的程序。本章节介绍停止LED闪烁的程序。

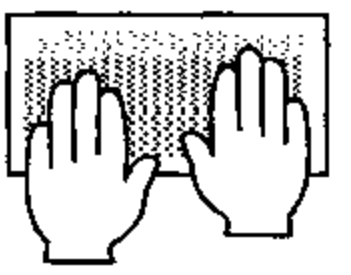
若要突然停止LED的闪烁，只要停止来自袖珍计算机的信号即可，非常简单。这里介绍LED闪烁缓慢停止的方法以及与以前不同的闪烁。

学习C语言的乐趣，在于稍用技巧即可得到有趣的程序。另外，本节末附有用BASIC语言进行控制的程序。其程序编制成与本书中的C语言程序几乎相同的控制形态。

学习BASIC语言，也借鉴学习C语言的方法。

I 利用拨钮开关关掉(OFF)LED的闪烁

下面来试拨钮开关ON时LED闪烁停止、拨钮开关OFF时LED闪烁再度开始的程序。



用拨钮开关切换控制装置时，要将C口的第0位置“ON”。即将下述语句写入程序。

```
outport (2,0x01);
```

练习题3.1

●编一段程序，让圆形排列的24个LED顺序闪烁，方向为顺时针旋转，速度按timer()函数120的快慢，并重复演奏斗来米发音阶，用拨钮开关ON中止LED的闪烁，用拨钮开关OFF再让LED闪烁起来。

```
/ *M·W·R·K.7c*/
main()
{
    int i,p,pl,sw;          ····
    clrscr()。
    start()
    while(1)
    {
        for(p=0x04;p<=0x10;p <<= 1)
        {
            pl = (p|0x21);
            outport (2,pl);
            for (i=0x01; i <= 0x80;i <<= 1)
            {
            if (inport(0))
            break;
            outport(1,i);
/*          disp(i);*/
            timer(120);
          }
          outport(1,0);
        }
    }
}
/* start() */                                   :初始设定函数
start()
{
        outport (3,0x90);
        outport (2,0x3e);
}
/* timer () */                                  :定时函数
timet(t)
int t ;
{
    int k;
    for (k=0; k<t ;k++);
}
/*  disp() */                                   :柚珍计算机显示函数
disp(d)
int d;
{
   int i;
   printf("%2x",i);
}
```

变量p为A.B.C3个口切换用，
变量p1对应各口的控制装置。

参见表11(p.42),亦可这样写

```
sw = inport (0);
if (sw == 1){ break;}
```

(a) if()语句

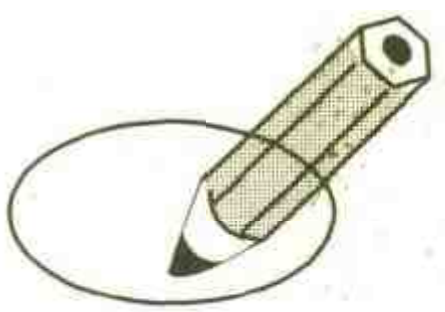
拨钮开关起关断作用。inport(0)的值为1时LED灭,inport(0)的值为0时,与未按拨钮开关的状态相同,LED继续回转

0x21 的含义

0x21 的含义，遵从表 1.11(p.42)如下：

拨钮开关侧装置切换 ON…0x01
+) 喇叭侧装置切换 ON…0x20
0x21

此状态下变量 p 的变化取 0x04、0x08、0x10 三种状态。这是 24 个 LED 分成 3 组 8 个 LED 1 组 1 组顺序进行切换。因而变量 pl 的变化是：



0x21	0x21	0x21
+) 0x04	+) 0x08	+) 0x10
0x25	0x29	0x31

因此，LED 顺时针旋转闪烁，且奏出音阶。
打开(ON)拨钮开关，LED 的闪烁立刻停止。

Ⅱ “圆圆”的减速

“圆圆”停止时，能不能使其更安静、更平稳、更优雅地停止？只要在定时函数的自变量上下功夫好像总能办到。

练习题 3.2

●编一段程序，让圆形排列的 24 个 LED 顺序闪烁，方向为顺时针，速度按 timer()函数 120 的快慢，反复奏 do re mi 音阶，通过开(ON)拨钮开关使 LED 的闪烁渐渐减速至停止，再通过关(OFF)拨钮开关让 LED 的闪烁动起来。

```
/*M·W·R·K.8c*/
main()
{
    int i,p,pl,j;
    clrscr();
    start();
    while(1)
    {
        for(p=0x04;p<=0x10;p<<=1)
```



```
        {
            p1 = (p|0x21);
            outport(2,p1);
            for(i = 0x01;i<= 0x80;i<<= 1)
            {
            outport(1,i);
            if (inport(0)) timer(j++);
            else                    拨钮开关一开(ON),j 的值不断 + 1,
            {
                timer(2);           拨钮开关关(OFF)时,LED 按 timer(2)
                                    的速度继续闪烁。
                j = 0;
            }
            disp(i);
        }
      outport (1,0);
    }
  }
/* start() */
start()
{
    outport (3,0x90);
    outport (2,0x3E);
}
/* timer () */
timer(t)
int t;
{
    int k;                          对定时器下边夫定时器的极限
    for (k = 0;k<t * 5;k++);
    if(t>= 32)
    {
        while(inport (0))
```

```
        {
            gotoxy(1,2);printf("SW 复位");
        }
    }
    clrscr();
}
/* disp() */
disp(d)
int d;
{
    int i;
    printf("%2x",i);
}
```



Ⅲ C 编程的其他

练习题 3.3

●编写"圆圈"反时针方向旋转的程序。

```
/* text7.c */
main()
{
    int i,p,p1;
    clrscr();
    start();
    while(1)
    {
        for(p = 0x10;p>= 0x04;p>>= 1)
        {
            p1 = (p|0x21);
            outport(2,p1);
            for (i = 0x80;i>= 0x01;i>>= 1)
```

```
                {
                if (inport (0))
                break;
                outport(1,i);
                 disp(i);
                timer(80);
                }
                outport(1,0);
            }
        }
}
/* start() */
start()
{
    outport (3,0x90);
    outport (2,0x3e);
}
/* timer () */
timer(t)
int t;
{
    int k;
    for (k = 0;k<t;k++);
}
/* disp() */
disp(d)
int d;
{
    int i;
    printf("%2x",i);
}
```



练习题 3.4

●编写 LED 两两闪亮的程序

```
/* text8.c */
main()
{
    int i,p,p1,j;
    clrscr();
    start();
    while(1)
    {
        for(p=0x04;p<=0x10;p*=4)
        {
          p1=(p|0x21);
          outport(2,p1);
          for(i=0x01,j=0x02;i<=0x80,j<=0x80;i*=2,j*=2)
          {
            if(inport(0))
            break;
            i=i|j;
            outport(1,i);
/*           disp(i);*/
            timer(80);
          }
          outport(1,0);
        }
    }
}
/* start() */
start()
{
    outport(3,0x90);
    outport(2,0x3e);
}
```

```
/* timer () */
    timer(t)
int t;
{
    int k;
    for (k = 0;k<t;k++);
}
/* disp() */
disp(d)
int d;
{
    int i;
    printf("%2x",i);
}
```

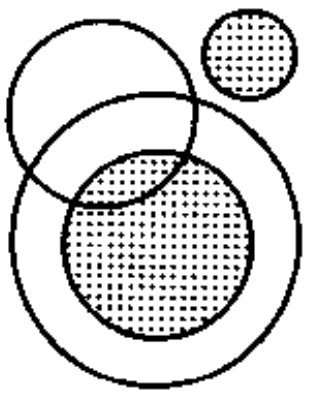


练习题 3.5

●编写 LED 随机闪烁的程序。

```
/* text10.c */
main()
{
    int i,p,p1,j;
    clrscr();
    start();
    while(1)
    {
        for(p = 0x04;p<= 0x10;p = 1)
        {
          p1 = (p : 0x21);
          if(inport(0)){break;}
            outport(2,p1);
            for(i = 0x01;i<= 0x80;i = 1)
            {
```

```
            i = i : 0x07;
            outport(1,i);
            timer(20);
          }
            for(i = 0x80;i>= 0x01;i>>= 1)
            {
                i = i - 7;
                outport(1,i);
                timer(20);
            }
            outport(1,0);
          }
    }
}
/ * start() * /
start()
{
    outport (3,0x90);
    outport (2,0x3e);
}
/ * timer () * /
timer(t)
int t;
{
    int k;
    for (k = 0;k<t;k++);
}
```

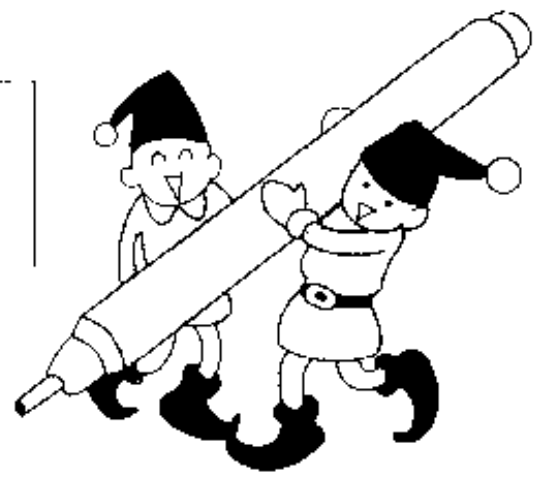


Ⅳ BASIC 语言的控制

仅用 C 语言的编程稍显不足，因此这里来试试让“圆圆”动作的 BASIC 语言的编程。

程序执行时，袖珍计算机的屏幕上即显示：

```
x x * 圆圆 * * *
0:QUIT 1:PB-LED 2:声音 3:转轮
4:拨钮 SW
```



其中，0：程序结束

1：初等生"圆圆"的 LED 的闪烁

2：初等生"圆圆"的 LED 和音阶

3："圆圆"的旋转

4：拨钮开关对"圆圆"的控制

输入各自的序号，确认程序的工作。

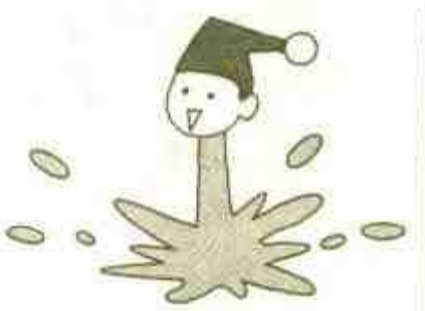
```
10 ' x x 圆圆 * *
20 OUTPORT 3,&H90
30 OUTPORT 2,&H3F
40 CLS:PC = 0:PB = 0
50 PRINT " x x x 圆圆 * * * "
60 PRINT "0:QUIT 1:PB-LED 2:声音 3:转轮"
70 PRINT "4:拨钮 SW"
80 K$ = INKEY$ :IF K$ = " THEN 80"
90 IF K$ = "0" THEN 150
100 IF K$ = "1" THEN 190
110 IF K$ = "2" THEN 340
120 IF K$ = "3" THEN 500
130 IF K$ = "4" THEN 820
140 BEEP 1:BEEP0:GOTO 10
150 OUTPORT 1,0
160 OUTPORT 2,&H3E
170 CLS:PRINT " x * 您辛苦了 * * "
180 END
190 'PB-LED
200 CLS
210 PRINT " x * * PB-LED * * x "
220 PRINT "请按 PUSH SW"
```

```
230 PRINT "用[0]回车"
240 PC = PC AND &HFE
250 OUTPORT 2,PC
260 PA1 = INPORT(0):PC = PC OR 1
270 PC = PC OR 1
280 K$ = INKEY$ :IF K$ = "0" THEN 40
290 OUTPORT 2,PC
300 OUTPORT 2,&H3E
310 PA2 = INPORT(0)
320 OUTPORT 1,PA2
330 GOTO 260
340 '
350 CLS
360 PRINT " * * * 声音 * * * "
370 PRINT "用[0]回车"
380 OUTPORT 2,&H3E
390 FOR I = 0 TO 7
400 Y = 2^I
410 K$ = INKEY$ :IF K$ = "0" THEN OUTPORT 2,&H3E:GOTO 40
420 OUTPORT 1,Y
430 FOR T = 0 TO 1000:NEXT
440 NEXT I
450 FOR I = 7 TO 0 STEP - 1
460 Y = 2^I
470 OUTPORT 1,Y
480 NEXT I
490 GOTO 380
500 'MAWARU
510 CLS:CLEAR
520 RPINT"圆圆的轮盘回转"
530 PRINT"用[0]回车"
540 EE = INT(800 * RAN#) + 100
550 P = 2
560 FOR K = 1 TO 3
```



```
570 K$ = INKEY$ : IF K$ = "0"THEN OUTPORT 2,&H3E:GOTO 40
580 P = P * 2
590 P = P OR &H20
600 OUTPORT 2,P
610 DT = 1
620 DT = DT OR DD
630 S = 1
640 OUTPORT 1,0
650 OUTPORT 1,S:' OR &H55
660 K$ = INKEY$ : IF K$ $ = "0" THEN OUTPORT 2,&H3E:GOTO 40
670 FOR T = 0 TO DT:NEXT
680 S = S * 2
690 IF S>128 THEN OUTPORT 1,0:GOTO 730
700 '
710 DT = DT + 10:DD = DT
720 IF DT<= = EE THEN 650 ELSE 760
730 NEXT K
740 GOTO 550
750 '
760 OUTPORT 1,S
770 FOR W = 0 TO 2000:NEXT W
780 OUTPORT 2,&H3E
790 OUTPORT 1,0
800 PRINT "END"
810 GOTO 40
820 'toggle sw
830 CLS:CLEAR
840 OUTPORT 2,&H3E
850 PRINT " * * 请 ON-OFF 拨钮 SW * * "
860 OUTPORT 2,&H01
870 T = INPORT (0)
880 LOCATE 1,1:PRINT"拨钮 SW 的状态为";T
890 LOCATE 1,2:PRINT"PUSH 0 RETURN"用按"0 "回车
900 K$ = INKEY$ : IF K$ = "0" THEN 40
```

```
910 OUTPORT 2,&H02
920 IF T = 1 THEN OUTPORT 1,&H0F
930 IF T = 2 THEN OUTPORT 1,&HF0
940 IF T = 3 THEN OUTPORT 1,&H00
950 GOTO 860
```



V 主程序的说明

10～40 行:初始设定

50～60 行:袖珍计算机的屏幕显示

70～140 行:处理内容的选择

0 号:程序结束　150～180 行

1 号:1 排 8 个 LED 的控制　190～330 行

2 号:音阶声音　340～490 行

3 号:让“圆圆”成为赌轮盘　500～810 行

4 号:用拨钮开关的 ON、OFF 控制 1 排 8 个 LED 的闪烁变化　820～950 行

VI “圆圆”赌轮盘的程序说明

540:产生随机数

580:定义 P=4,8,16(&H04,&H08,&H10)

590:圆形排列的 LED 的 8 位口与音阶装置的连接

610:定时器初始设定为 1

620:定时器 DT 和 DD 不作“或”时,定时循环返回时不减少。

630:移动的初始设定

650:LED 和音阶发生

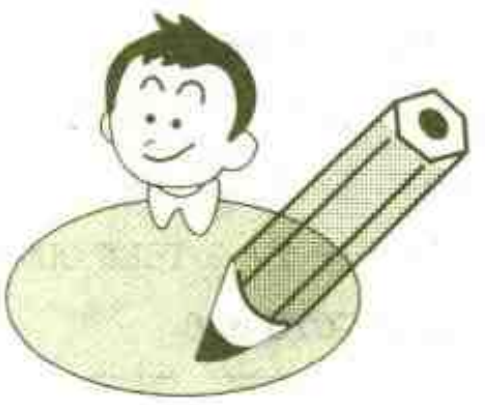
670:定时

680:向左移动

710:定时的瞬时值

720:定时值的累积与随机数数量的比较

760:停止时 LED 亮
770:2000 次循环间,LED 闪烁和音阶发生
790:OFF



Ⅶ 拨钮开关的 ON、OFF

860:切换到拨钮开关
870:读取输入



致 读 者

"圆圆"和C语言编程的学习如何?
学过一通后,不是有如下的感觉吗?

袖珍计算机有趣!
C语言是便利的语言!
控制,一试很简单!

体验过"圆圆"控制的各位与本书的读者,以及观看过控制录像带的各位:

进一步做一些不同的事情试一试!
例如:①做一些与此不同的其他控制
② 用非C语言的语言编程
③ 更深入地运用C语言
④ 制作40针卡边的接口板等等

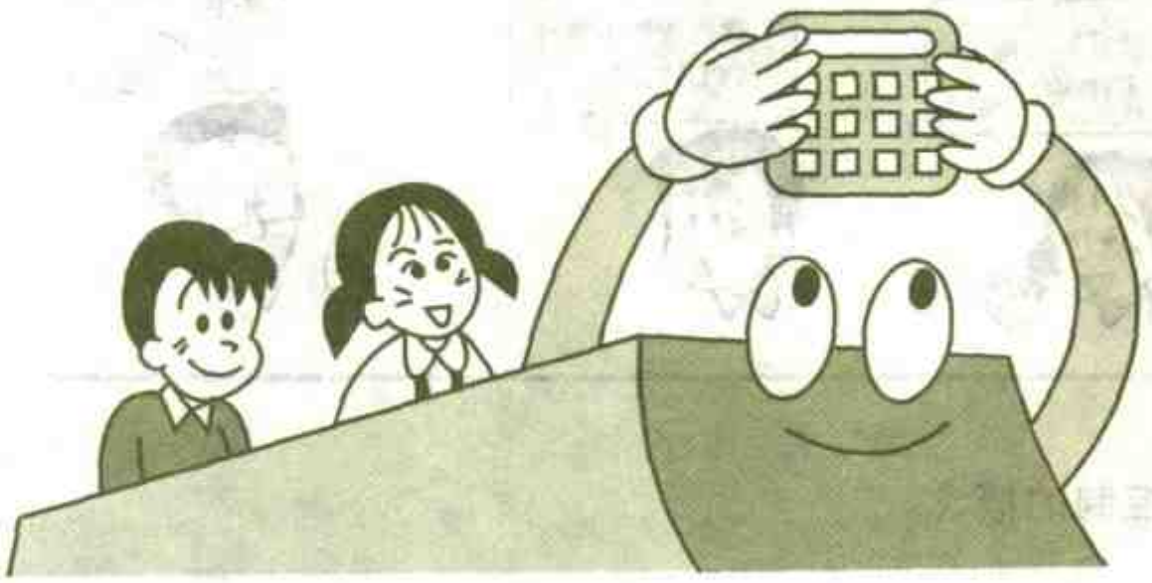
如能萌生一些新鲜的感觉,定能增强实践的动力。

相信许多人(哪怕是1个人),对于自己的与前不同的发现都会感到异常喜悦。

小野上冢 纯夫

第3章

计算机控制基础



本章学习内容

计算机控制需要分几个步骤进行。每一步的操作并不困难,只要实际操作过,很快就能自如地运用。每一步骤的内容将在第4章按顺序介绍。

本章在学习每一控制步骤之前,先来学习使用单板机的计算机控制基础知识。

例如,想用微机控制机器人时,要按什么顺序进行准备?控制时都需要些什么工具?了解这些即是学习本章的目的。

另外,也要加深理解计算机控制的基本术语。例如,在第4章说明ROM写入器的使用方法,但在本章首先要介绍何为ROM写入器,为什么要使用它。

同样也要介绍理解有关CPU板及程序调试等有关知识。

本章还在充分理解微机控制的顺序概要之后,要对各个步骤进行练习。这样一来,即使是初学者也能轻松愉快地掌握微机控制。

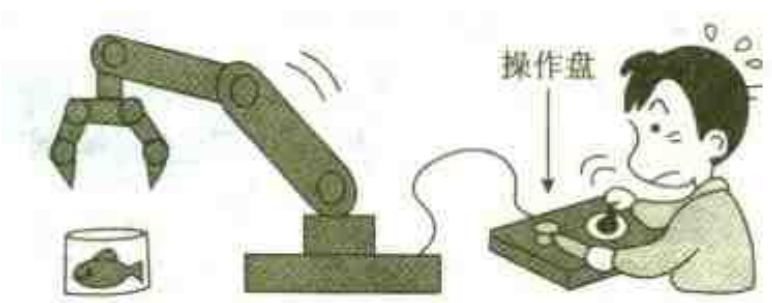
1 微机控制的种类



微机控制的概念

从现在开始学习微机控制，什么是微机控制呢？

假如，要用机械臂从右向左移动罐头，这时就有由人来操作机器人的操作盘使机械臂移动的方法。

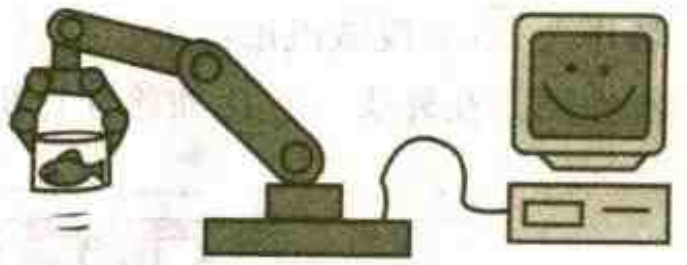


(人工操作)

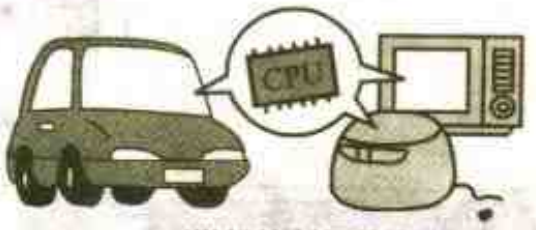
那么，有必要练习机器人操作盘的使用方法，不是吗！

是的。而且，想要机械臂动作时，总要有人来操作。但是如果使用计算机并能将动作的指令传给机械臂的话，就可以自动地控制机械臂。并且，使用计算机一般比人更能正确地操作机械臂。这种使用计算机(微机)对某种对象进行自动控制就称之为微机控制。

用微机控制时，想要改变机械臂移动的方向时，怎么办呢？



(微机控制)



(微机控制很普遍)

只要改变计算机的程序，无论怎么控制机械臂都行。这里，举的是机械臂的例子，实际上除此之外，微机控制还可以用在许多地方。

听说家里的微波炉啦、空调啦等等电器中都装有微机控制。

汽车发动机控制以及安全装置中也利用微机控制。生活中这种利用微机控制的产品很多。

控制用微机

已经明白使用计算机的控制叫做微机控制，可是计算机和微机有什么区别吗？

虽然都叫计算机，但大小不同。研究所等用的超级计算机自不必说，一般家庭开始普及的个人计算机以及掌上电脑也都是计算机的一种。这些计算机中，小型的叫做微机。而用微机对某种控制对象进行控制即为微机控制。



虽然都叫计算机但种类很多

用于控制的微机都有哪些呢？

用于微机控制的计算机分为以下三种。

(1) 个人计算机

个人计算机有台式及笔记本式之分。其优点是操作方便、程序编制及更改容易，但是由于尺寸大而无法装入机器内部，价格也贵。

(2) 袖珍计算机

能够装入口袋那么大的计算机称为袖珍计算机。最近，许多工业技校的学生都在购买个人用袖珍计算机。袖珍计算机价格便宜、操作简单，常常用来进行微机控制。但是，装入机器还仍显过大。

(3) 单板机

在一块印制电路板上构成的简单的计算机称为单板机。单板机常用来学习计算机的基本功能。而且，专门用作微机控制的非常小型的(印制电路板只有

名片大小)单板微机市面上也有出售。这种控制用单板机不带键盘及显示器,元器件都露在外头。它比袖珍计算机价格更低,由于其尺寸小适合装入机器内部。





使用个人计算机及袖珍计算机的控制中,要另外准备称为接口板的电路。而在控制用单板机中,接口电路是预先内置其中的,因而可以一体使用。但是,初次处理好像较难,初学者以先不用为宜。



装入机器中的微机控制部分,使用控制用单板机是最适宜的啦。





是的。但是很遗憾,很多人因为在观念上的先人为主,觉得单板机难用而一般不太使用。即使在工业技校的微机控制课程中,一般也使用袖珍计算机。理解控制用单板机的优点、掌握其使用方法,是本书的目的。

例如,在本书后部讲到的机器人相扑比赛中,机器人的重量规定最大不超过3kg。有此限制,想制作强有力的机器人时,就只能使用控制用单板机。



一旦掌握使用方法，使用控制用单板机的控制也一定非常有趣。

用于微机控制的 CPU



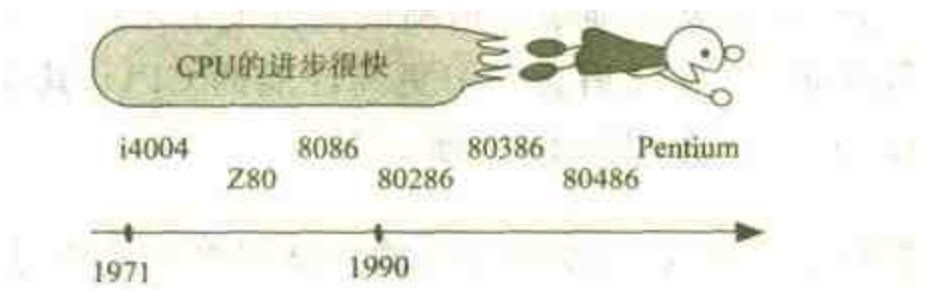
你们知道 CPU 吗？



知道。CPU 就是 Central Processing Unit(中央处理装置)的缩写。它是计算机中进行运算及控制的重要核心部分。



是的。说 CPU 的进步就是计算机的进步一点也不过分。从 1971 年美国 Intel 公司推出 i4004CPU 开始，许多厂商陆续推出了多种 CPU。现在成为主流的高性能的 CPU 有 80486、奔腾、Power-PC 等。





微机控制中也用这种高性能的 CPU 吗？

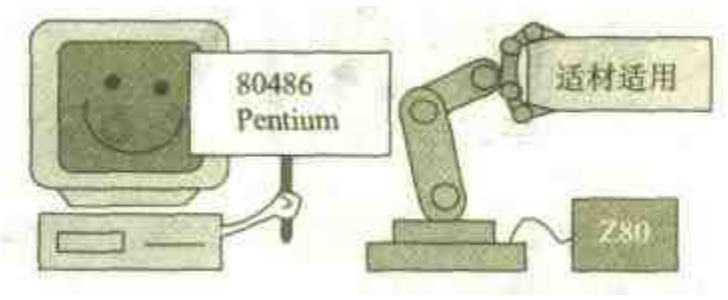
高性能 CPU 的最大特点是处理速度极快。回想一下前面的机械臂的例要让机械臂作眼睛都跟不上的高速运动，即使从机械臂的性能来讲也是不可能的。



总之，一般的微机控制中，现在成为主流的超高速的 CPU 并不必要吧。



是的。微机控制中经常使用的 CPU 是 Z80 及 M6809。Z80 是美国的 Zilog 公司 1975 年开发的 CPU，用于现在的个人计算机力量不足，但用于通常的微机控制还是绰绰有余。





Z80 是 20 多年前推出的 CPU，现在还能方便地得到吗？



不必担心。现在日本的许多厂商还在制造出售具有与 Z80 同样规格的 CPU。虽然 Z80 开发的相当早，但在微机控制领域，即使现在，Z80 还在积极地工作。采用 Z80 的控制用单板机市面上销售的也很多。所以，本书介绍

使用 Z80 的微机控制。





使用很早以前开发的 CPU 来学习不会白费劲吧？



绝对不会。从理解 CPU 的观点来考虑，了解了 Z80 就了解了 CPU 的基本东西。与一开始就立即学习最新的高性能的 CPU 相比，还不如在切实了解 Z80 这种简单的 CPU 之后再学习更高性能的 CPU，其学习效果更好。总之，Z80 的了解是了解新型 CPU 的基础。



后面的课程，是使用采用 Z80CPU 的控制用单板机学习一些控制方法吧？





是的。有关要用的控制用单板机，放在后面说明。控制用单板机上没有键盘及显示器等，所以在控制的准备阶段，要借用个人计算机的帮助。当然最终还是只用一块控制用单板机，来实行对对象机器的控制。



练习题

1. 用于微机控制的典型计算机都有哪些种类？
2. 使用控制用单板机都有哪些优点？

解答

1. 个人计算机、袖珍计算机、单板机。
2. 小型、可装入机器内部、廉价、接口电路内置。

2 微机控制的实例



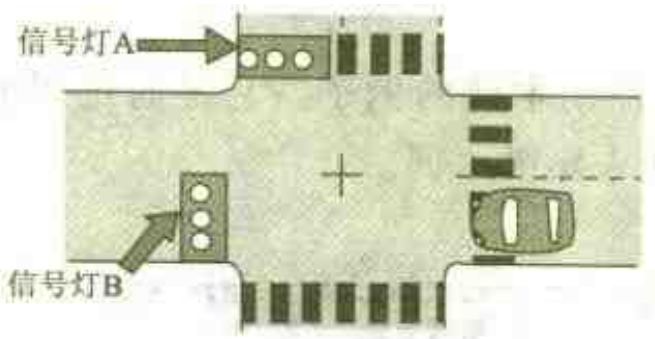
信号灯的控制

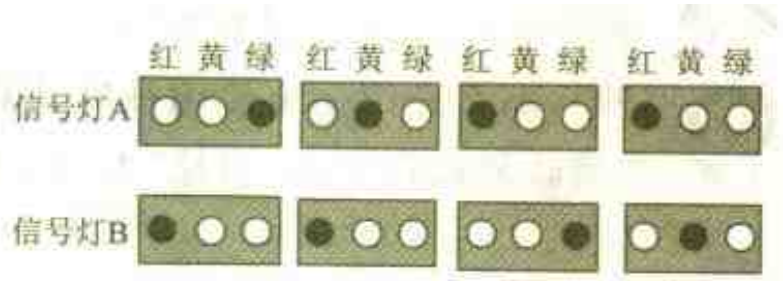


试想一下十字路口。指挥交通用信号灯 A 与 B 的工作情况是怎样的呢?



在第一组中信号灯 A 为亮绿时,信号灯 B 必须亮红。出于这种考虑,信号灯 A 与 B 的工作状态对应如下:





信号灯的工作



要让这些信号灯工作可以利用微机控制。



右图为使用个人计算机对信号灯的模型进行控制的例子。



信号灯模型控制

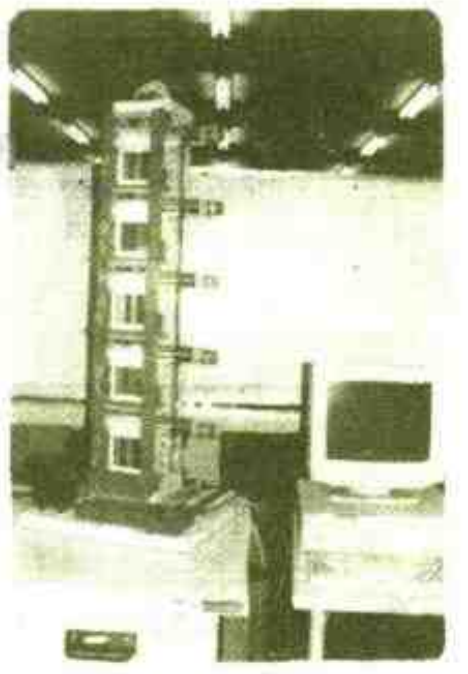


电梯示意图

电梯的控制

下边来看电梯的运行。假设一层楼的商场中有一部电梯，现在电梯停在 1 楼，2 楼有客人想下 1 楼，他按了叫电梯的按钮。紧接着 3 楼也有客人想下 1 楼，也按了按钮。

此时，尽管 2 楼先叫，电梯还是通过 2 楼运行到 3 楼，之后再停在 2 楼。它没有让想下到 1 楼的 2 楼客人，乘上狭窄的电梯一直到 3 楼，因而这种运行方法对于客人想得很周到。考虑到各种各样的情形，要让电梯运行得恰当，客人等待的时间要短。


电梯模型的控制

看起来很普通的电梯，实际上进行相当复杂的运行。

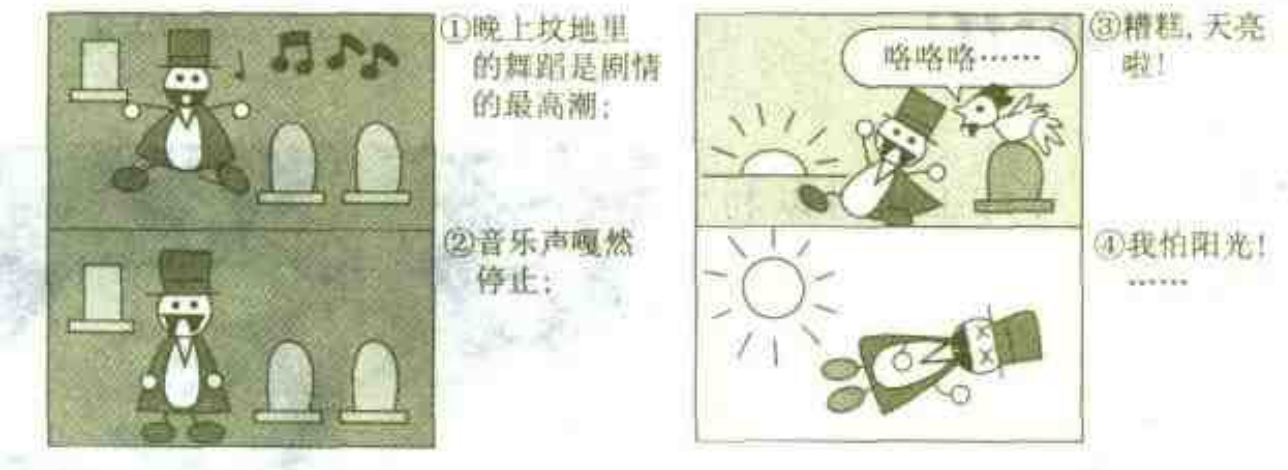
像电梯运行那样，按预先确定的顺序来动作的控制称为程序控制。

微机控制也适于下面的情形。右图为控制电梯模型的例子。

木偶的控制

已经看过了信号灯及电梯等实用的微机控制的实例，请再介绍一些微机控制娱乐设施的例子吧。

好吧，那么介绍一个使用袖珍计算机控制木偶的例子。它是通过用电动卷筒收放系在木偶“多拉”上的 5 根线来操纵木偶。



木偶戏的故事情节

只是简单地让木偶作动作太没意思啦，所以要给木偶的动作配上故事情节。

木偶"多拉"的故事情节

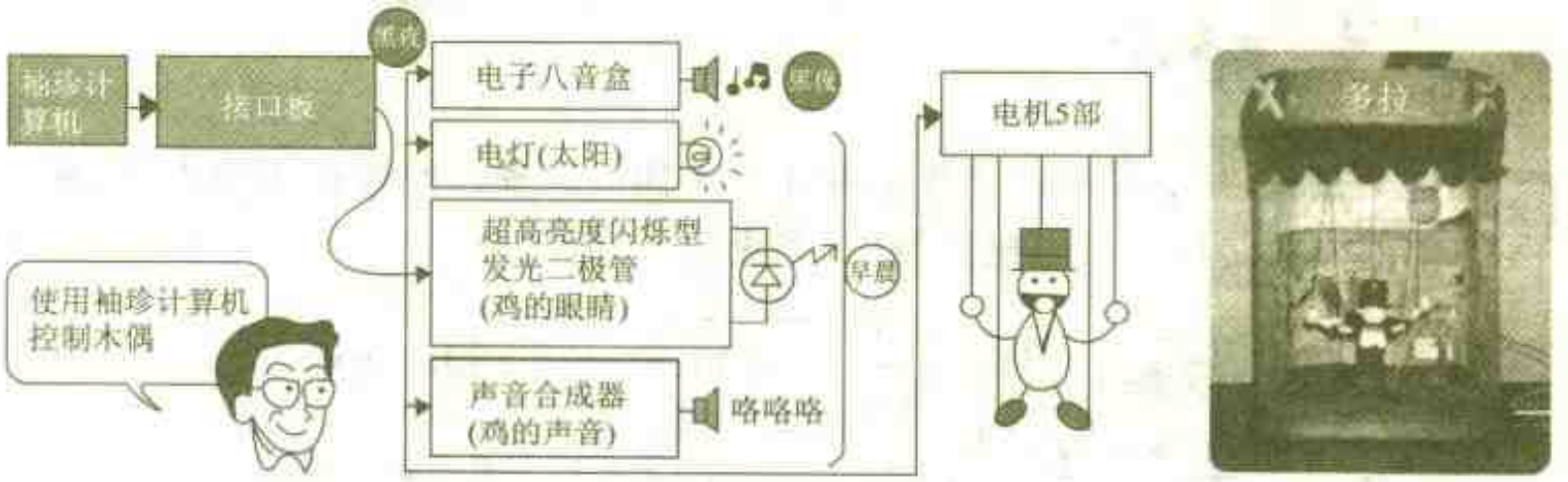
①合着音乐(电子八音盒)的节拍,"多拉"快乐地跳着舞;
② 音乐停止;
③ 太阳开始升起(电灯亮),早晨来到。公鸡醒来了(装在眼睛里的发光二极管闪烁),开始报晓(声音合成器);
④ "多拉"倒在地上,死去了。



吸血鬼"多拉"怕太阳光。



这个木偶的例子中,是用袖珍计算机控制着电机、电子八音盒、电灯、声音合成器、以及 LED(发光二极管)。



皮耶罗魔术师的控制



再介绍一个有趣的微机控制的例子。它可以揭露皮耶罗木偶的魔术秘密。

皮耶罗魔术师的故事情节

① 皮耶罗一拿起箱子,台子上什么也没有;
② 放下箱子,再拿起时,台子上出现胡萝卜。皮耶罗的身后露出兔子的脸;
③ 放下箱子,再拿起时,台子上出现鱼。皮耶罗的身后露出猫的脸;
④ 放下箱子,再拿起时,台子上出现香蕉。皮耶罗的面孔变成猴子脸。

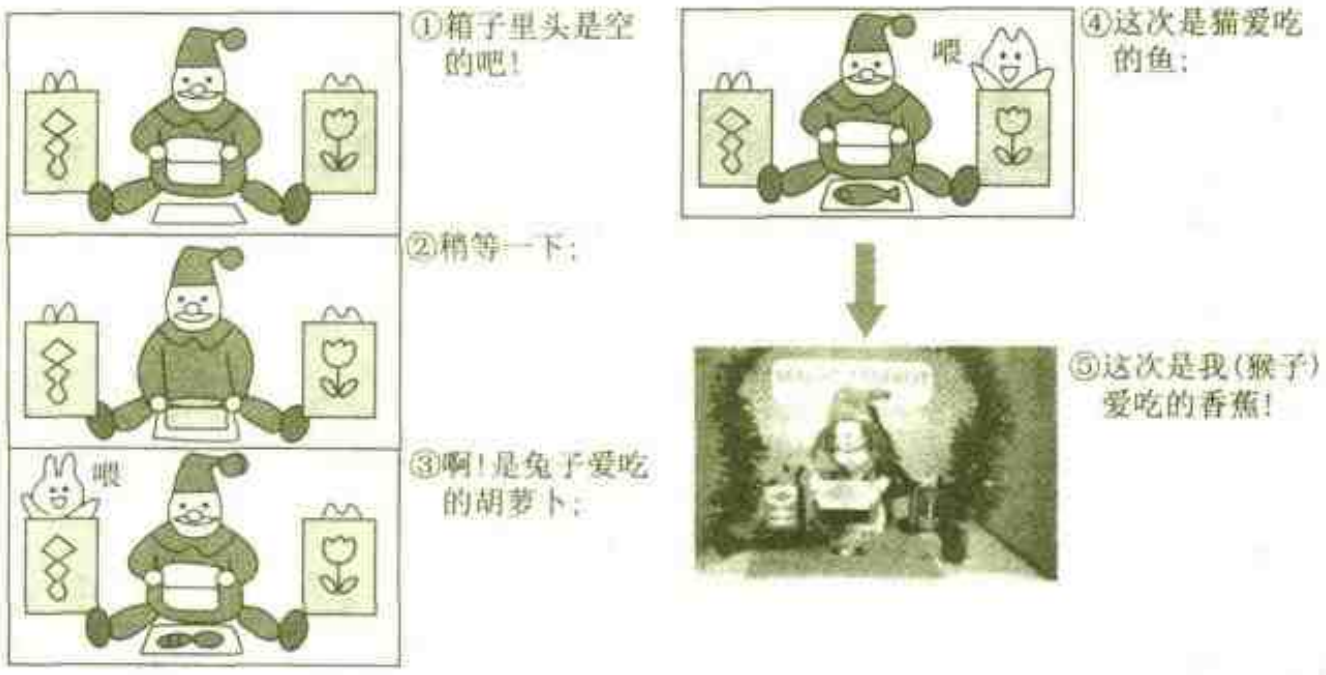
以上故事情节,随着皮耶罗的饶舌(声音合成器)而展开。



"饶舌"线操木偶戏,好像很有趣唉!

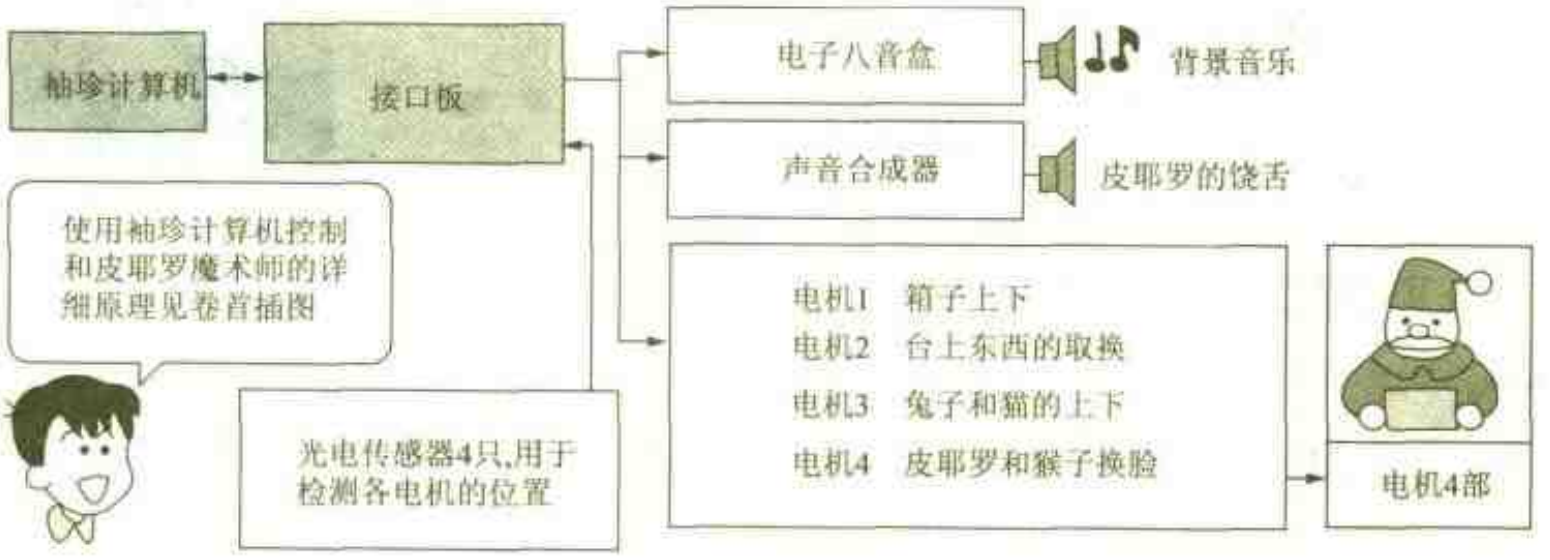


什么机关能让台子上出现胡萝卜、香蕉什么的？





这个机关，用的是200多年前制作的木偶的原理。原理说明略去，但是台子的机关是用一部电机控制的（参见卷首插图）。



自立型相扑机器人的控制



最近，机器人相扑比赛大会呼声高涨。这是让装有控制用单板机的小型机器人在直径154cm的摔跤场上进行相扑比赛。有关此大会的详细情况在第4章进行介绍。



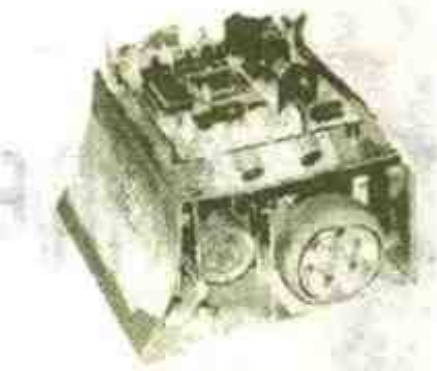
我们制作的机器人大力士也去参加相扑比赛吗？



是的。自己做的机器人大力士比赛时，特别兴奋。对于机器人大力士，有大小及重量的限制，最好的办法是安装小型的单板机。

为对驱动机器人大力士的电机、识别摔跤场边框白线的光传感器、寻找比

赛对手的超声波传感器等诸多电路进行控制而利用微机。有关机器人的制作方法，在第 4 章进行具体介绍。



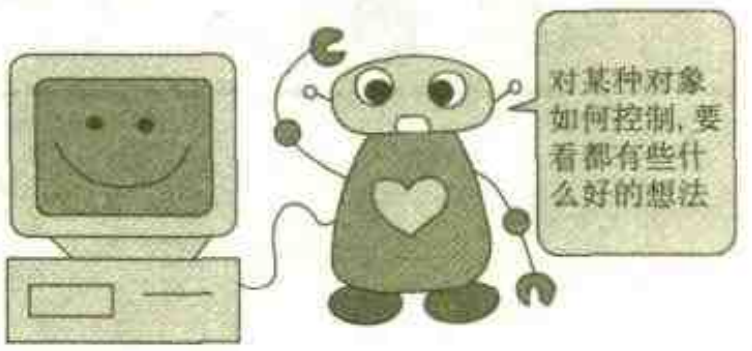
相扑机器人

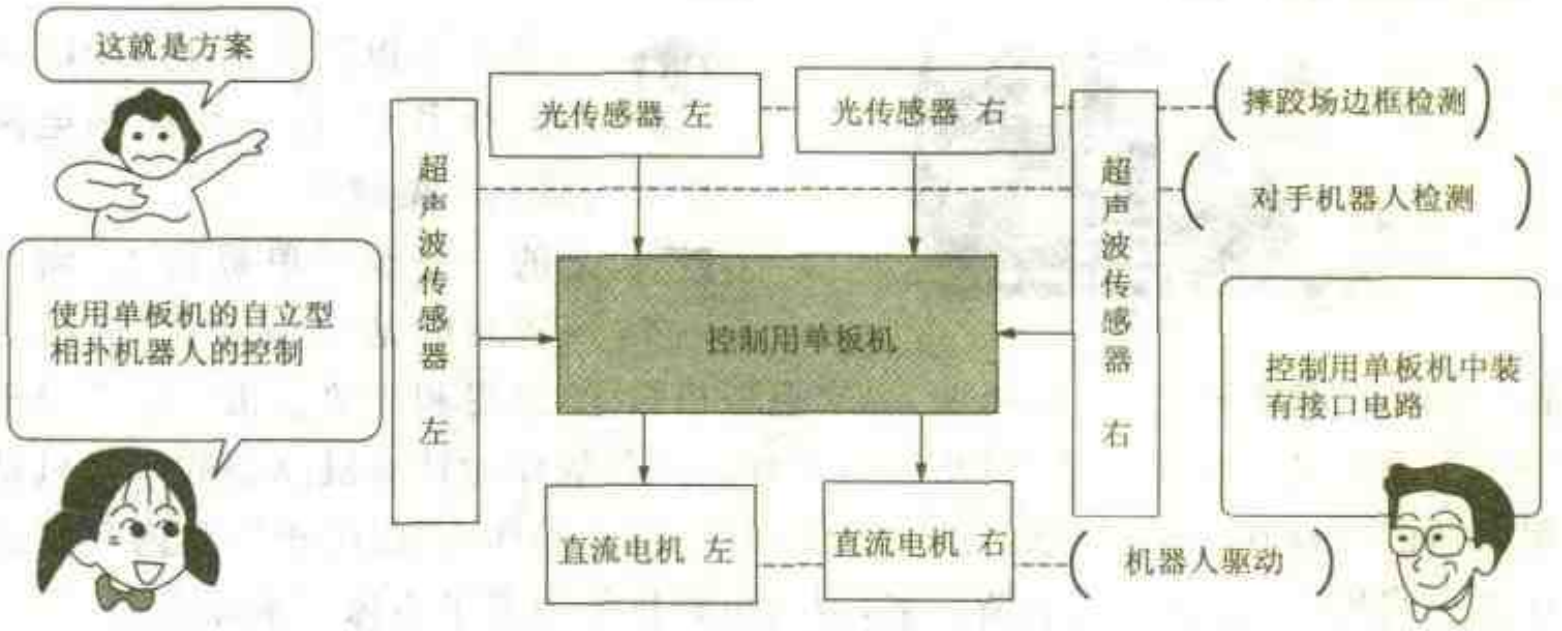


前面已经见过微机控制的例子，都很有趣。想自己也实验一下看看。但是好像很难，稍微有些担心。



微机控制的基础部分一点不难。按照顺序进行简易的说明，所以不必担心。如果有利用微机对某种对象如何进行控制的想法思路，大家就一定能做出愉快的独创性的作品。





练习题

考虑一下，使用个人计算机、袖珍计算机、单板机的控制都有些什么不同。

解答

无论使用哪种控制用计算机，其控制原理完全不变。仅仅是准备方法及控制电路不同。另外，根据是否要否装人控制对象内部决定使用哪种微机。

3 微机电路板



单板机

前面对单板机已作过简要介绍，还记得吗？

记得。单板机就是把 CPU 及存储器 IC 装在一块印刷电路板上构成的小型微机。


单板机

是的。一般的单板机上，输入装置装的是小型键盘，输出装置装的是 7 段 LED。20 多年前，要掌握微机技术，总得利用单板机。这是由于要学习硬件和汇编程序语言(机器语言)时，简单的作为计算机原型的单板机是很合适的。现在，计算机已经高度进步，计算机自身作为“黑匣子”(其内部是些什么另当别论)要处理的事情很多，因而单板机的需要量不像以前那么多。

微机控制是与硬件密切相关的技术，所以，用单板机进行学习，对于掌握微机控制当然是很有利的。

是这样的。一定要找机会学学单板机。肯定会加深你对微机的理解。

以汽车驾驶为例，即使不知道发动机及煞车的机理也能驾驶。但是，知道了这些以后再驾驶，就能够按照机械能力及特性进行非常合适的操作。而且简单的故障自己也能修理。微机也与此相同。在理解了基本原理的基础上再来操作，不仅能够更好地发挥微机性能，而且对于各种故障也能有一定的对策。

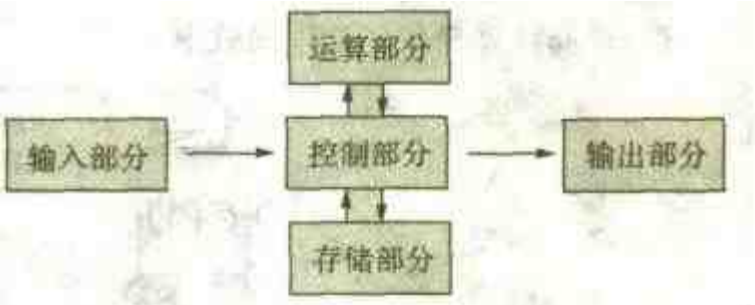


请对单板机也稍微做一些说明吧。

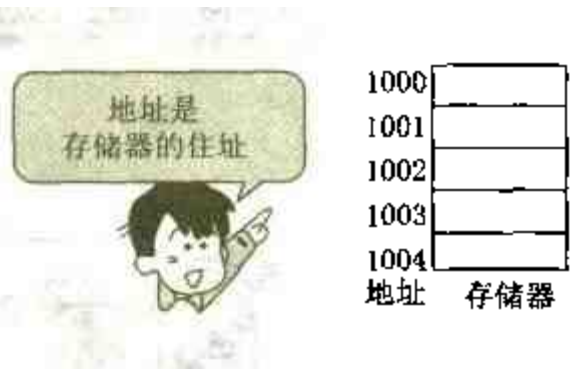
好的，知道啦。一般的单板机的基本构成如下：程序用机器语言（十六进制数）从键盘输入，还要指定输入存储器的地址。

存储器的地址是什么？

例如，使用个人计算机输入 BASIC 语言程序时，无需特别了解其程序存放在计算机存储器上的哪个部分。这是由于 BASIC 系统是自动来决定输入程序的存放场所的。但是，机器语言中，输入程序的存放场所是必须由编程来决定的。存储器中的程序及数据的存放场所，是带着地址管理的。即，指定了地址就能指定存储器上任意的存放场所。



单板机的基本构成



机器语言处理是不是很麻烦？

的确如此。与高级语言（BASIC 等）相比处理上稍显麻烦。但是，机器语言具有高速、细微处理的优点。

高级语言编成的程序在单板机上不能用吧？

利用个人计算机，有办法将 BASIC 及 C 语言编写的程序转换，可在单板机上使用。关于这一点，在第 4 章介绍。

现在还有很多厂商在出售教学用单板机。从组装单板机开始，能够更有效地对计算机进行学习。

Z80 家族

本书中提到的 Z80CPU，配有与其组合使用的几种专用 IC。这些被称为 Z80 家族。顾名思义，Z80CPU 是这个家庭的顶梁柱，支持它的周围的各

种 IC 是这个家庭的成员。

Z80 家族

① Z80CPU 是作为微机关键的 CPU。

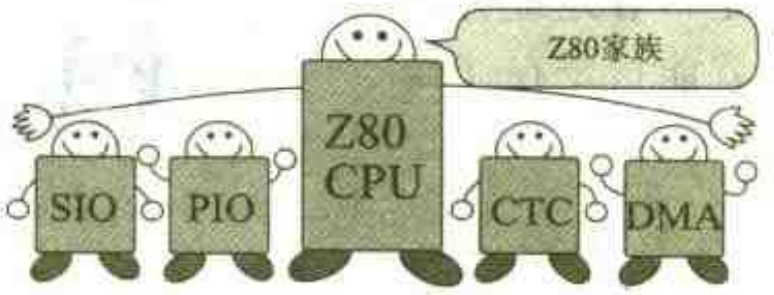
② Z80PIO PIO 是 Parallel Input Output 的缩写。是给微机并行输入数据、从微机并行输出数据时所必需的接口 IC。

③ Z80SIO SIO 是 Serial Input Output 的缩写。是给微机串行输入数据、从微机串行输出数据时所必需的接口 IC。

④ Z80CTC CTC 是 Counter Timer Circuit 的缩写。常常要利用反复处理来制作任意的定时器。CTC 就是起定时作用的 IC。让 CTC 进行多次重复出现的处理，可减轻 CPU 的负担，提高计算机整体处理的效率。

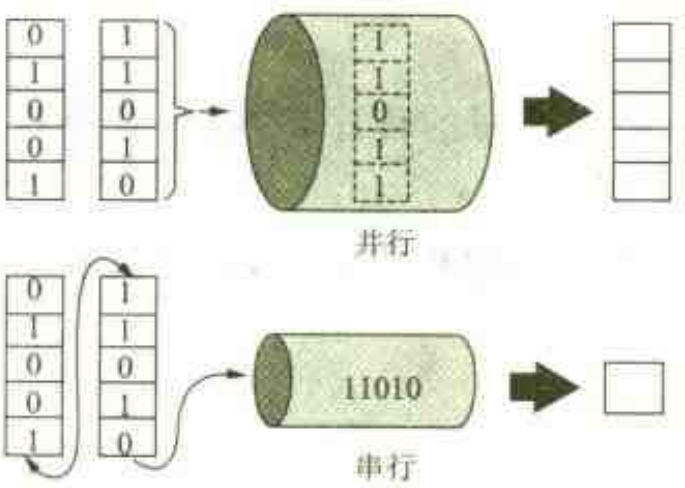


⑤ Z80DMA DMA 是 Direct Memory Access 的缩写。使用 DMA 技术，无需 CPU 介入，即可在存储器与 I/O 接口间直接进行数据传送。





所谓并行、串行是什么？





这指的是传送数据时的方式。所谓并行，是将整理好的数据并排一次传出的方式。而所谓串行，则是将数据一个一个连成串传出的方式。

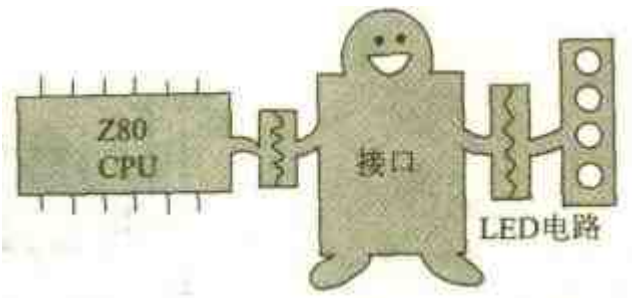


并行方式可以在短时间内传送很多数据，但需要许多信号线。而串行方式只需一根信号线即可，传送数据时稍微费点时间。

所谓接口是什么?

将不同的功能连接一起使用时,其起连接作用的部分称为接口。

例如,想使用 Z80CPU 控制 LED 的闪烁时,就要在 CPU 和 LED 间使用接口 IC。Z80 家族 IC 设计得非常适于相互间组合使用。



PIO 中,8255IC 是很有名的哟。

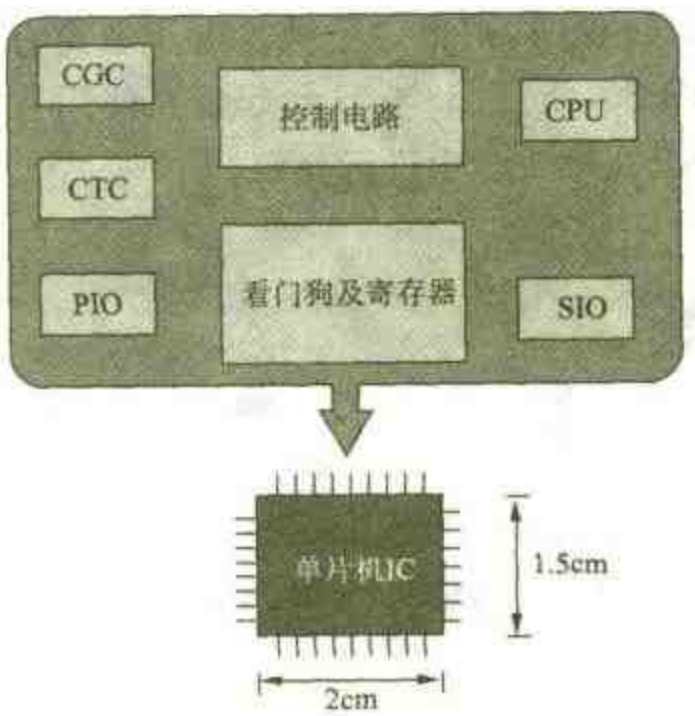
是的。尽管 8255 不属 Z80 家族,但它很便于使用,也常与 Z80CPU 组合使用。但是在中断功能方面,仍然是 Z80 家族中的 Z80PIO 是最适宜的。

单片机 IC

单板机与个人计算机及袖珍计算机相比,尺寸小、重量轻,且适于装入机器内部。按照将单板机用于控制目的,把更为小型的称为控制用单板机。控制用单板机中,没有输入装置及输出装置,使用的 IC 也是特殊的。

特殊的 IC 是什么东西?

控制用单板机中,使用的 IC 称为单片机 IC。前面对 Z80 家族作过简单的说明。单片机 IC 即是将 Z80 家族 IC 及其他功能件封装于一只芯片中的小型 IC。



单片机 IC 中,集成了很多功能。

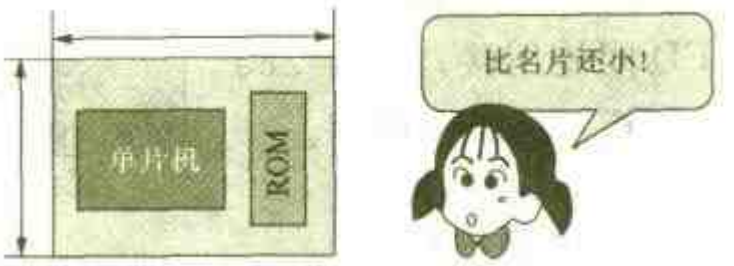
单片机 IC 中,除封有 CPU、PIO、SIO、CTC 外,还含有 CGC(时钟发生电路,Clock Generator Circuit)、看门狗以及寄存器等。

CGC:时钟发生电路
看门狗及寄存器:检测 CPU 的误动作、回到正常动作的电路。

微机板



按照用途在单片机 IC 上连接存储器 IC(ROM 及 RAM)以及若干逻辑 IC,即可当作漂亮的微机来使用。这就是控制用单板机。

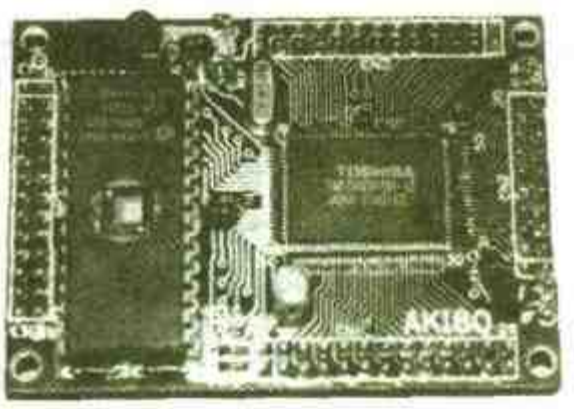




使用单片机 IC 的控制用单板机,比微机教学用的单板机更小。



而且一开始就装有利于微机控制的许多功能。总之,作为装入小型机器的微机来用是最适宜的。





是这样的。对控制用单板机的使用方法进行简易的说明是本书的目的。控制用单板机,也简单地叫做单板机或微机板。本书在此后就将控制用单板机称为微机板。

使用含有 Z80CPU 的单片机 IC 的微机板,许多公司都有销售。其基本的使用方法都一样。但在本书中,为了更为具体地进行说明,则集中于秋月电子通商有限公司出售的 AKI-80 微机板进行讲述。AKI-80,便宜(3800 日元起价)、使用方便,是首选的微机板。关于 AKI-80 的详细情况在第 4 章中进行介绍。

练习题

1. 对单片机 IC 进行说明。
2. 适于装入小型机器的微机是哪种类型?

解答

1. 参照本节。
2. 微机板。

4 微机控制的顺序

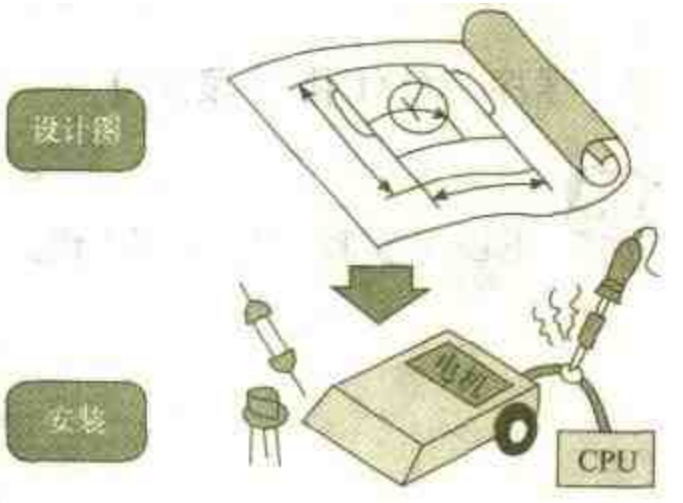


相扑机器人的微机控制

请教我们使用微机板进行微机控制时的实际顺序吧。

好。那么以自立型相扑机器人为例来考虑一下微机控制的顺序。

首先要按照设计图制作相扑机器人的硬件部分，组装电机驱动部分及传感器控制部分等的电子电路，再与单板机相连接。



但是，如果只完成硬件部分，这个相扑机器人并不会动。还必须完成判断决定如何动作的程序，即软件部分。在这里说明制作此软件部分的顺序。



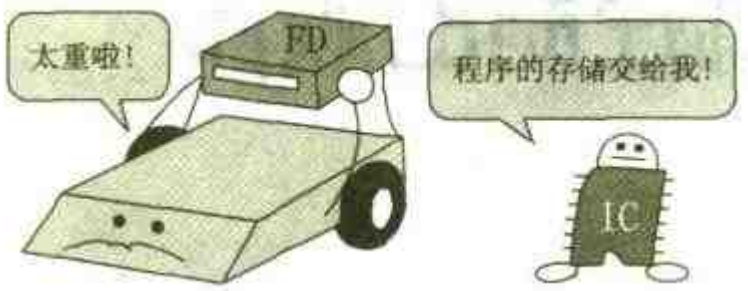
总之，就是编写控制相扑机器人的程序并将其装入实际的相扑机器人中的顺序。

所谓将程序装入相扑机器人，是怎么回事？

微机的程序并不是有形的东西，而是数据的集合。这些数据通常存储在软盘或硬盘

上。


可是，将软盘驱动器或硬盘驱动器装在大小及重量都有限制的相扑机器人上是不行的。




是的。但是可以装在IC上。能够存储程序的IC叫做什么?


叫做存储器IC。存储器IC有数据可读可写的RAM(Random Access Memory)和只供读出的ROM(Read Only Memory)两种。切断电源，RAM存储的内容会消失，而ROM即使切断电源，存储的内容仍旧保持着。

- IC存储器
 - RAM
 - 读写两用
 - 电源断开，数据消失
 - ROM
 - 读出专用
 - 电源断开，数据仍保持

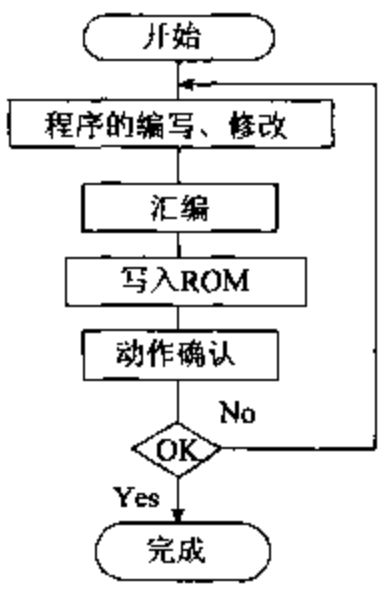

ROM


是的。微机板上，RAM和ROM两者都可安装。所以，程序是写入ROM而装上的。

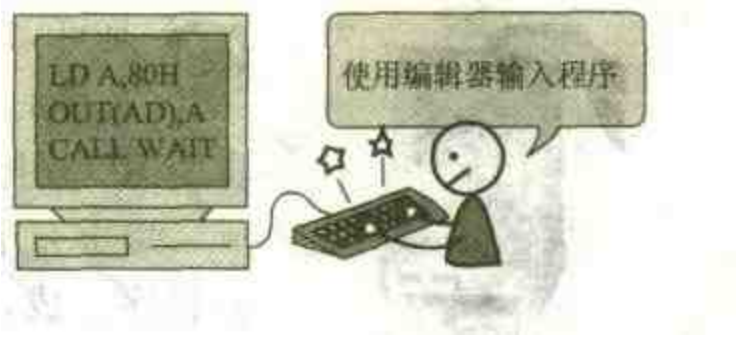
将程序写入ROM，称为程序的ROM化。

程序ROM化的顺序1


下图是实现程序ROM化的基本步骤。



程序ROM化的顺序1

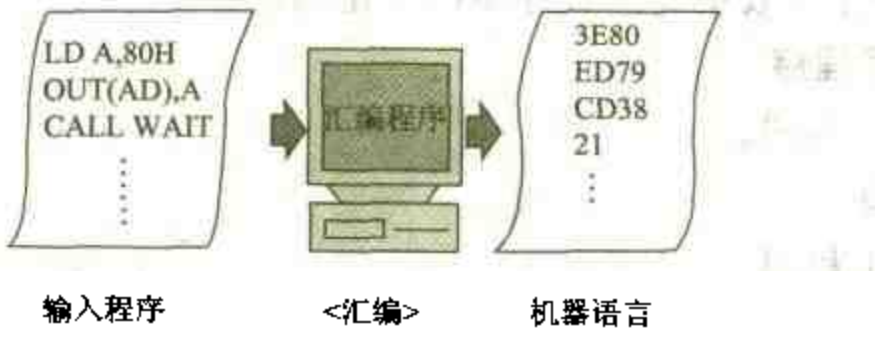


(1) 程序的编写

使用个人计算机，用 BASIC 或汇编或 C 语言编写程序。编辑程序的软件（编辑器）是必需的。编辑器可用一太郎等字处理器的软件代用。

(2) 汇　编

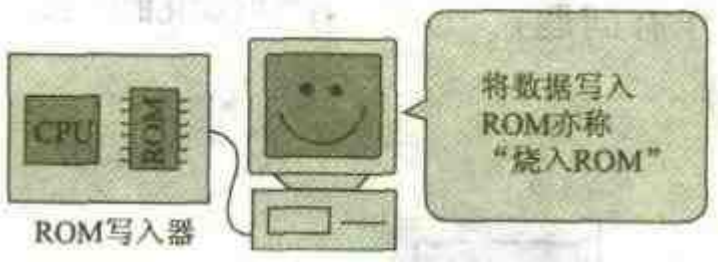
将写好的程序要变为 Z80 的机器语言。变为机器语言称为汇编，变换的软件称为汇编程序。此项工作使用个人计算机完成。



输入程序　　<汇编>　　机器语言

(3) 写入 ROM

需将 Z80 的机器语言程序写入 ROM。为写入需使用个人计算机和称为 ROM 写入器的装置。



(4) 动作确认

将写有程序的 ROM 插在微机板上，对相扑机器人的动作进行确认。若有问题时，回到(1)修正程序，然后再进行(2)、(3)、(4)等步骤。




程序 ROM 化的基本步骤已经明白了，相当麻烦呢。


程序中难免有故障，所以我想能按最初设想来动作的确很难。


一旦将程序写入 ROM，还能将修改后的程序再写入吗？

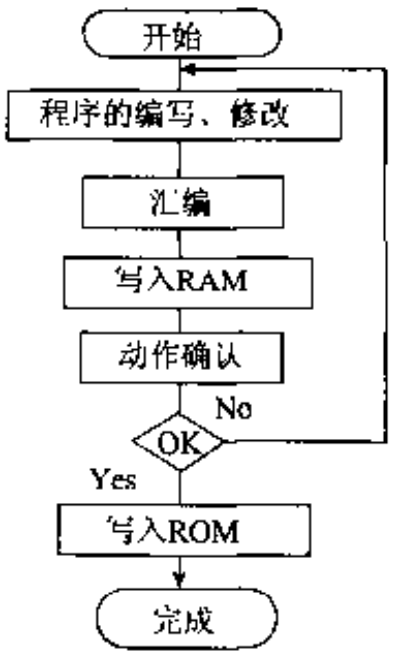


ROM擦除器

是可以的。但是，给已经写有程序的ROM再次写入程序时，首先要清除ROM中的原内容。对此需使用称为ROM擦除器的装置。

以上介绍的步骤，是程序ROM化的基本方法。但是，到程序完成，需要多次改写ROM，恐怕不下数十次。这样的话，工作效率太差。所以推荐下面介绍的“顺序2”。

程序ROM化的顺序2



程序ROM化的顺序2

下面来介绍效率高的程序ROM化的顺序。

(1) 编写程序

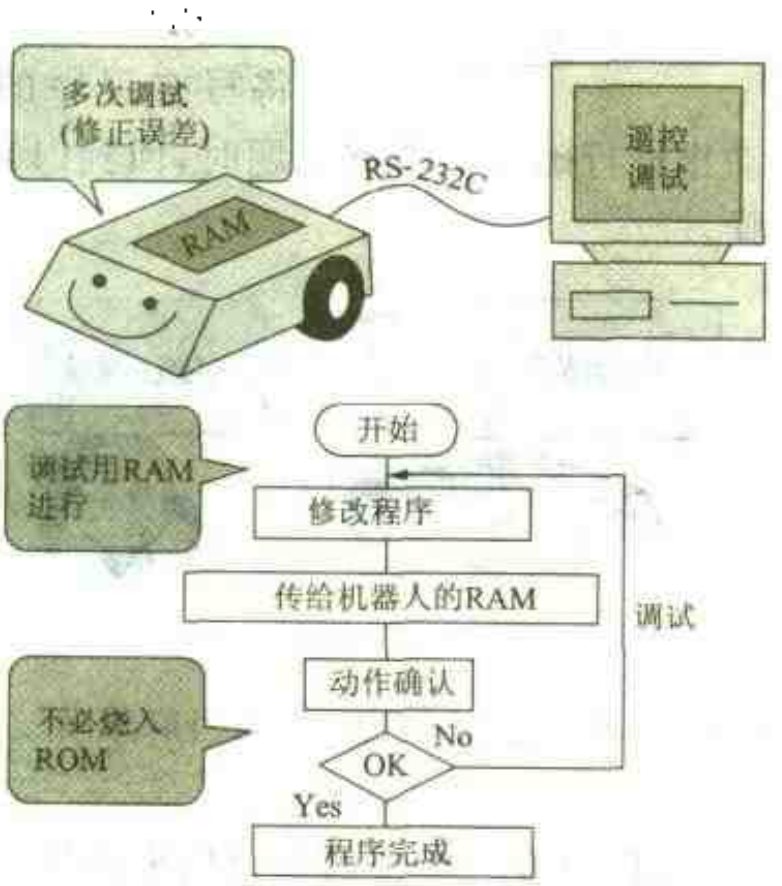
与顺序1相同。

(2) 汇编

与顺序1相同。

(3) 写入RAM确认动作

将Z80机器语言程序写入安装于相扑机器人微机板上的RAM中。为此，需在个人计算机上使用称为遥控调试的软件，如果相扑机器人的动作有问题，还要对程序进行修改。



(4) 写入 ROM

将经过步骤(3)确认的程序写入 ROM。写入时使用个人计算机和称为 ROM 写入器的装置的情况与顺序 1 相同。



利用微机板上的 RAM 对程序的工作确认后，再写入 ROM。这样一来，无需多次使用 ROM 擦除器和 ROM 写入器改写 ROM，因而效率较高。





本书介绍的是按此顺序 2 的程序 ROM 化方法。

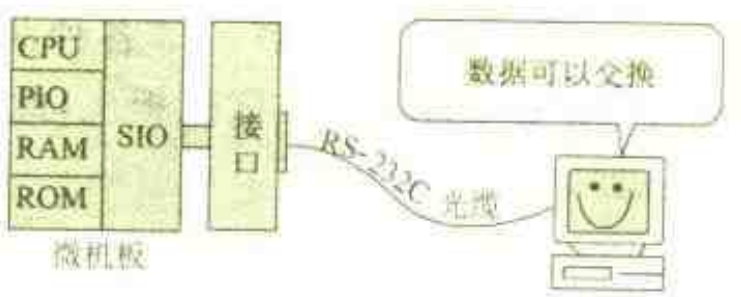
个人计算机与微机板间的通信



请讲一下有关将在个人计算机上编好的程序转送到微机板上 RAM 的过程。



在微机板的单片机 IC 中，内藏有称为 SIO(串行输入输出)的系列 IC 的功能。利用此功能，再添加若干外接电路，经由 RS-232C 光缆，可以实现微机板与个人计算机间的数据交换。





所谓 RS 232C 指的是串行数据传送规格。在个人计算机内部有 RS-232C 用的连接器。



都需要什么样的外加电路？

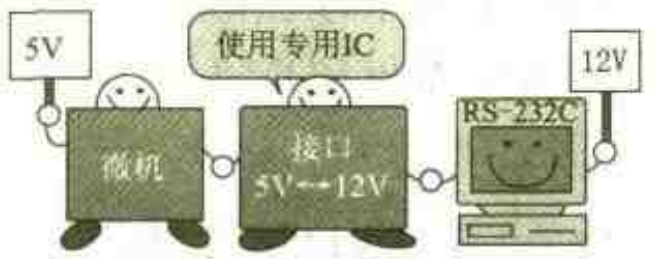


微机板用 5V 电源工作，数据的电压也要以此为准。而按照 RS-232C 规格，数据电压用的是 12V。所以，将来自个人计算机中 RS 232C 的数据电压降到 5V，或者将来自微机板的数据升压到 12V，都需要电平匹配电路。

对此，使用简单的专用 IC 市面有售，只需简单的电路即可。

电平匹配后，个人计算机和微机板间数据的交换，即通信的准备工作即告完成。接着就要运行个人计算机与微机板两者间通信用的程序。个人计算机

的遥控调试程序中含有通信程序功能，而微机板则要使用写有通信程序的ROM。





怎样准备写有通信程序的ROM？



通信程序含在遥控调试用的软盘中，使用ROM书写器可将其写入ROM。关于这一点下一章将具体介绍。





使用个人计算机能够在与微机板通信的同时，对微机板用的程序进行修改(调试)。



完成程序ROM化，必须要有个人计算机等各种装置以及软件的帮助。当ROM化完成后，若将此ROM装在微机板上，此微机板即可单独用来控制相扑机器人等对象机器。

练习题

试考虑程序ROM化的“顺序1”与“顺序2”各自的优点。

解答

顺序1：不需要遥控调试。

顺序2：程序的调试容易。

Q 挑战题页

① 对某种机械，试比较人工控制与微机控制。 ☞P94

② 在我们的身边，都有哪样利用微机控制的产品？ ☞P94

③ 微机控制都用些什么种类的计算机？ ☞P94

④ 试说明控制用单板机。 ☞P95

⑤ 微机控制中，为什么至今还在使用 Z80CPU？ ☞P97

⑥ 作为愉快的微机控制的例子，试想出一个好点子。

⑦ Z80 家族指什么？ ☞P106

⑧ 单片机 IC 是什么？ ☞P107

⑨ 什么是接口？ ☞P107

⑩ 试考虑微机板的优点。 ☞P108

⑪ 比较存储器 IC 中的 RAM 与 ROM。 ☞P110

⑫ 比较微机控制的两种顺序。 ☞P110

⑬ 什么是 ROM 写入器？ ☞P111

⑭ 什么是 ROM 擦除器？ ☞P112

⑮ 使用微机板时，为什么需要个人计算机的帮助？ ☞P111

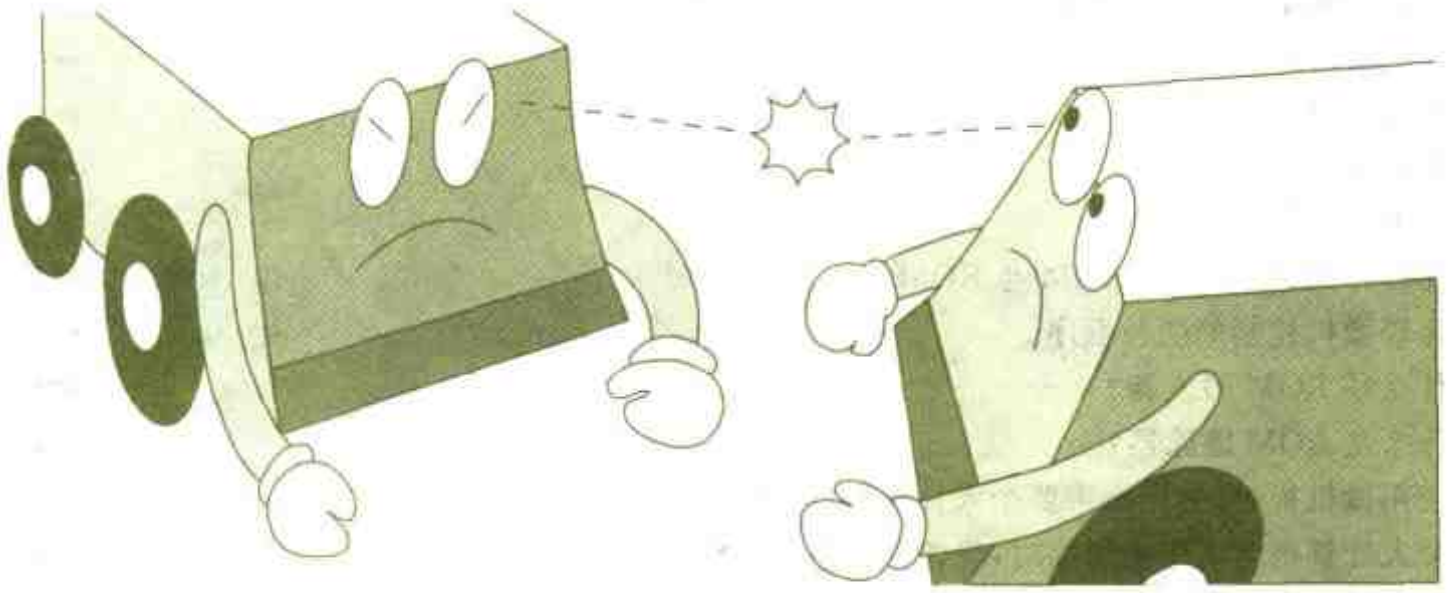
⑯ 个人计算机与微机板通信时，应该注意哪些事情？ ☞P113

⑰ 解释以下术语及词汇： ☞P113
编辑器、汇编、汇编程序、调试、烧入 ROM。

⑱ RS-232C 是什么？ ☞P113

第4章

机器人的制作方法



本章学习内容

第3章介绍了微机控制的基础，相信大家大概已经能够理解微机控制的概要以及控制工作的顺序。

微机控制，实际上是自己亲自手握电烙铁、操作个人计算机最初掌握的技术。并且，那怕有一次实际进行微机控制的经验，其后自然会渐渐应用开来。

本章，以相扑机器人的制作为例来介绍微机控制的实际情况。希望大家不要光读,建议实际试作。

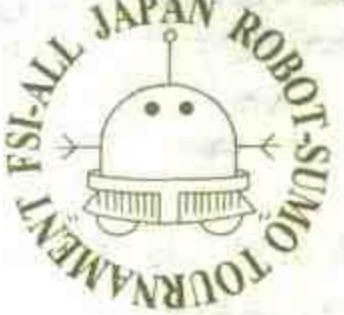
本章中提到的相扑机器人起名为"太郎"。首先介绍"太郎"的概要，很好地抓住了整体的情况后再开始实际学习。

"太郎"有一个简单的结构，但是根据不同的目的可以改良成更强的机器人大力士。这样，作成相扑机器人后，可以进行对战比赛。

对于初次经历微机控制的人来讲，说不定也有工作进展不顺利的情形。但不要停止，要努力做到最后。我相信,大家通过本书的学习所掌握的知识，将 来一定大有用处。

1 机器人相扑比赛大会概况





机器人相扑比赛大会



机器人相扑，作为富士软件株式会社 20 周年的纪念活动，于 1990 年 3 月举办了第一次比赛大会，共计有 147 台机器人大力士展开了角逐。这次大会赢得了很大的反响，之后，参加比赛大会的机器人数量逐年持续增长。



机器人大力士是不是一年比一年变强啊？



是的。现在先要在地方比赛大会中胜出，然后参加全国比赛大会。1993 年，又增设了高中生机器人相扑比赛大会，因而高中生参加比赛越来越容易，比赛大会也愈来愈引人关注。



大会场景1



大会场景2

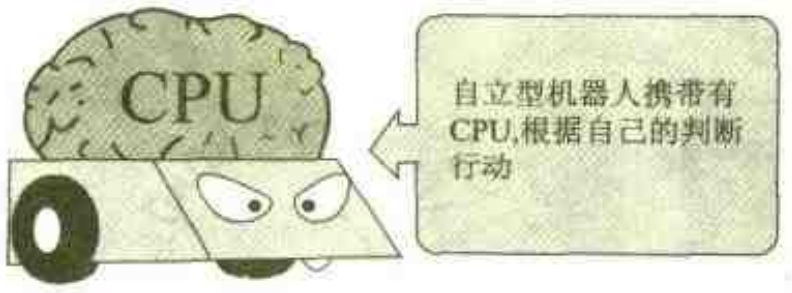
要制作相扑机器人，工业领域内广泛的知识与制造技术是不可缺少的，因而通过制作机器人正适合培养工业技校学生的动手能力。自己制作的机器人，

参加这种比赛会很有趣。所以，很多工业技校都面向比赛大会努力制作机器人大力士。

自立型和遥控型



机器人相扑比赛大会，设有自立型和遥控型两部分。



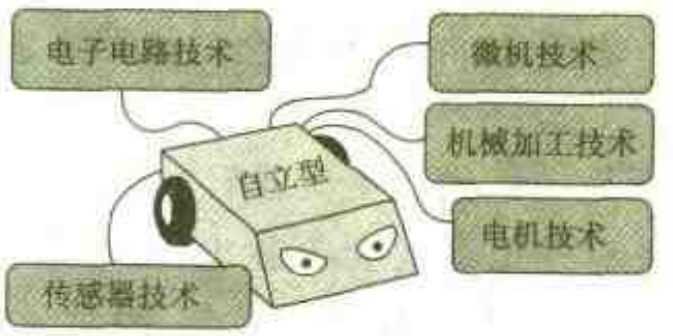
(1) 自立型

是指由所携带的微机进行控制的机器人。参赛者在比赛一开始按下机器人的开始开关后，只能在旁边观战。机器人还装有各种传感器，可寻找发现对方机器人进行攻击。

(2) 遥控型

是指参赛者用无线电控制进行操作的机器人。常使用市售的遥控器。为了取胜，需要增加机器人的功能，这会影响参赛者的操纵。由于对微机控制技术无特别要求，因而可以比较简单地制作机器人。







自立型相扑机器人更适合作学习微机控制的数材。



自立型相扑机器人的制作中，除微机控制技术外，电子电路技术、传感器技术以及机械加工技术等也是不可缺少的。

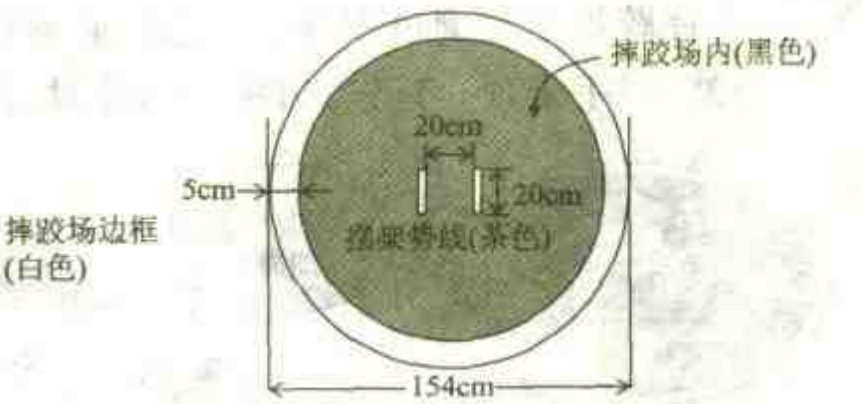


通过制作一台机器人大力士能学到许多工业技术呢。



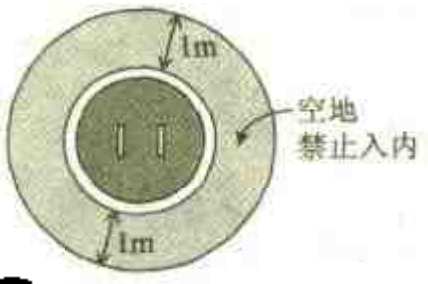
正因为如此，本书以自立型相扑机器人为对象，来介绍其制作方法。

比赛规则(自立型)



机器人相扑比赛,是在直径154cm的圆形摔跤场上进行。摔跤场的表面贴有黑油毡橡胶,摔跤场的边缘为白色,摆架势线为茶色。

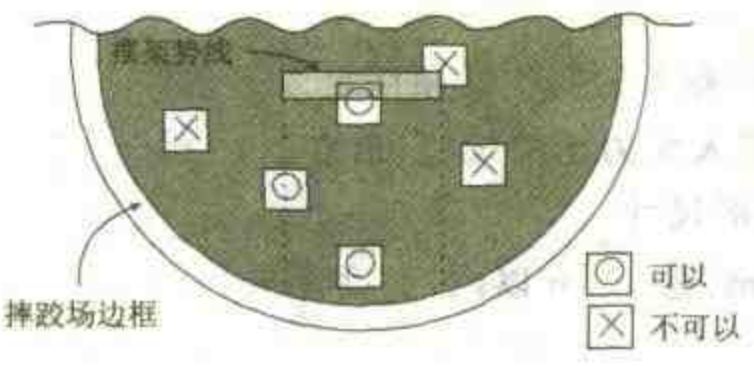
摔跤场周围留有1m的空地,目的是为了防止机器人携带的各种传感器的误动作等。参赛者将机器人先放在各自的摆架势线上,听裁判信号后,按开始开关。机器人必须在5秒以后才能开始动作。开始开关按下后,参赛者不得再接触机器人。





如果按下开始开关后,5秒之内机器人动作应视为什么行为呢?

这时就要警告。警告两次,给对方机器人加1分。面且,最初机器人的放置位置有如下的规定:



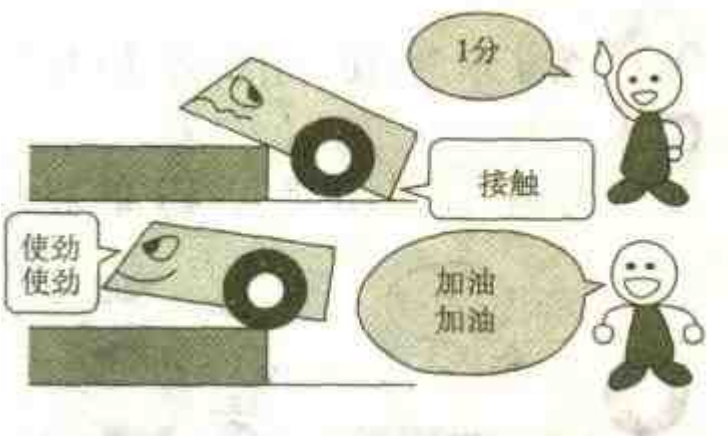
机器人所在位置

比赛时间为3分钟,三局两胜制。但是,超过了限制的3分钟时间后,先得1分者为胜。

怎么才算有效得分?

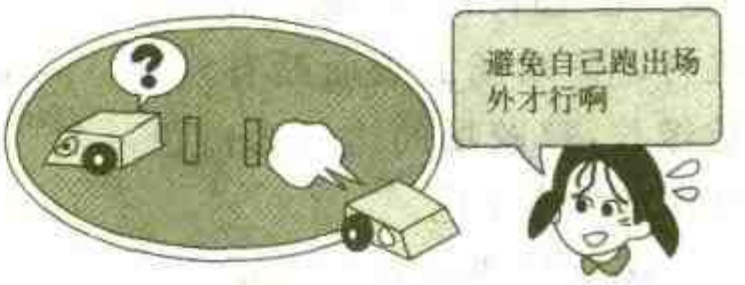
将对手机器人顶出摔跤场外得1分。就是说,必须使对手机器人身体的一部分触到摔跤场的外部才算。

就是说身体仅有少许露出摔跤场并不得分。



在摔跤场上翻倒也不算输。总之必须顶出摔跤场外才算。
当然，对手未顶，自己飞出摔跤场外时，也要给对手记 1 分。





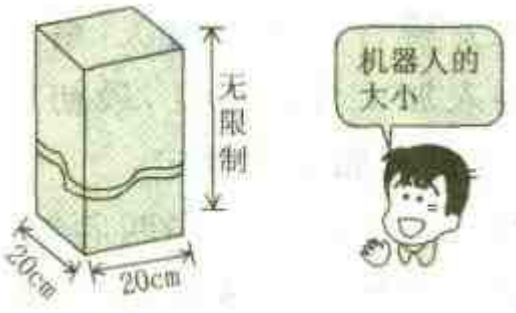
要想自己不跑出摔跤场外，至少要安装感知摔跤场边白线的传感器。

还有，一方有 1cm 以上零件脱落时，也要给对方记 1 分。





机器人大力士的规格



相扑机器人的制作必须满足以下条件。

1. 机器人大力士的规格(自立型)

(1) 外形尺寸

长 20cm、宽 20cm 以内，高度无限制。

(2) 重量

3000 克以内。



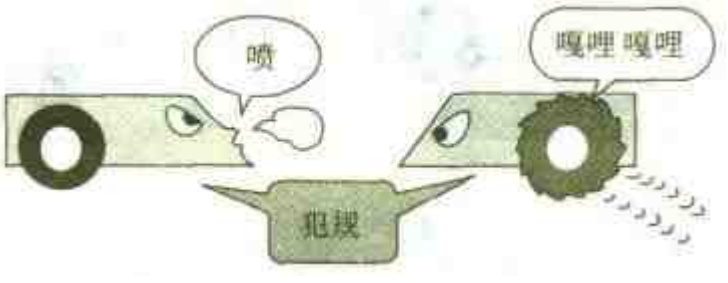
为了取胜，使用什么办法都行吗？

机器人的制作上有以下的禁止事项。

2. 制作上的禁止事项

① 不许故意破坏或污染摔跤场；

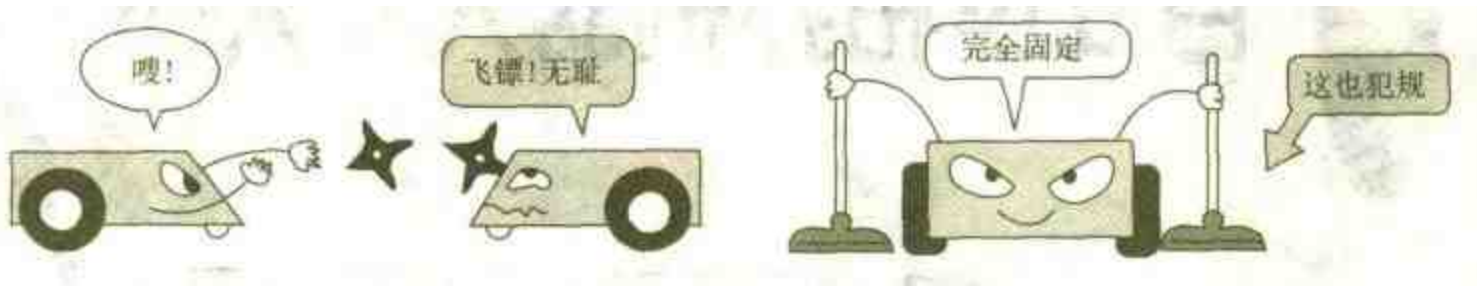
② 不得向对手吹喷液体、粉末及气体；



③ 不得内藏发火装置；

④ 不得内藏飞镖装置；

⑤ 不得用吸盘或浆糊将机器人完全固定在摔跤场上。



实验用摔跤场

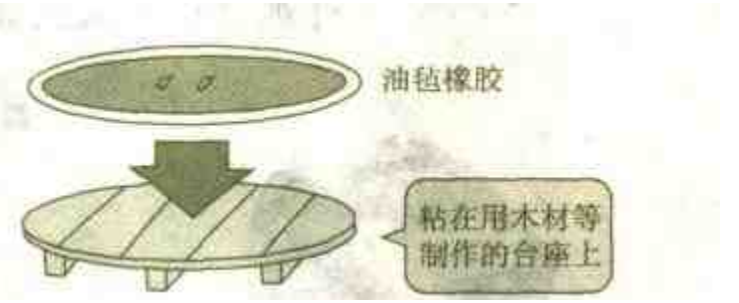


制作机器人大力士并使其动作时，实验用的摔跤场不可缺少。

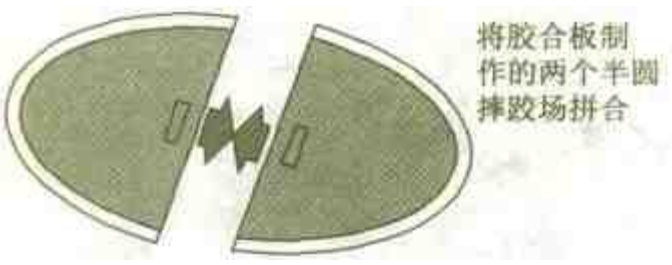


实验用摔跤场的制作

(1) 使用油毡橡胶



若要简单而有际地制作近似的摔跤场，需要购买油毡橡胶。富士软件株式会社的机器人相扑比赛大会委员会就有油毡橡胶出售，但是价格稍贵。将油毡橡胶粘在台座上制作摔跤场时，要十分注意不要在橡胶上出现凹凸不平。



(2) 使用胶合板

加工胶合板也能制作模拟摔跤场。这种材料便宜且容易得到，但大张的胶合板不易得到，因此要将两个半圆拼合成一个圆形。拼合连接时注意不要出现缝隙。再喷上油漆，简易的摔跤场就作成了。

练习题

1. 总结自立型相扑机器人的比赛规则。
2. 确认相扑机器人的制作规格。

(参照书中内容给出答案)

2 自立型相扑机器人“太郎”



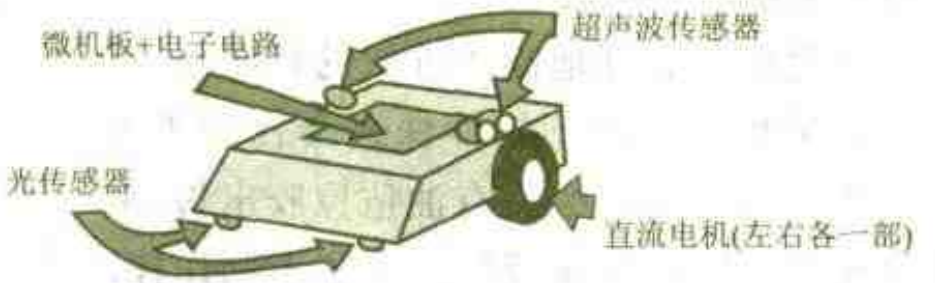
“太郎”的结构

在介绍机器人大力士“太郎”的制作方法之前，先来看一下“太郎”的整体情况。

“太郎”的外观及构成大致如下：



“太郎”



“太郎”的硬件，可分为以下几个部分来考虑。

“太郎”的硬件

(1) 机械部分

机体、电机、轮子

(2) 微机部分

Z80 微机板

(3) 电子电路部分

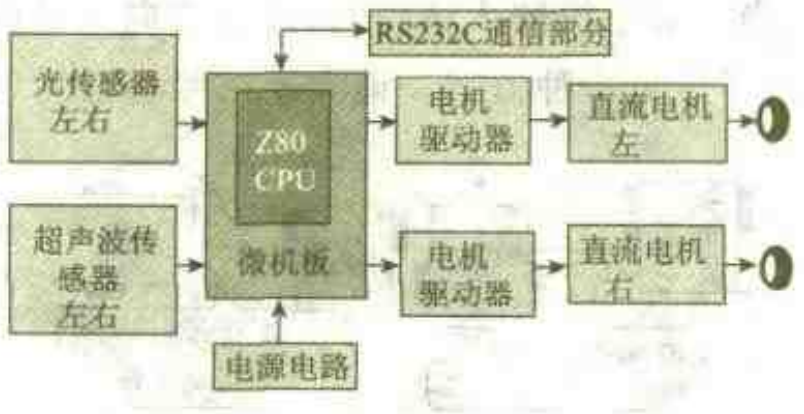
电源电路、电机驱动电路、RS-232C 通信电路

(4) 传感器部分

光传感器电路、超声波传感器电路



各部分以 Z80CPU 为中心，组成如下：



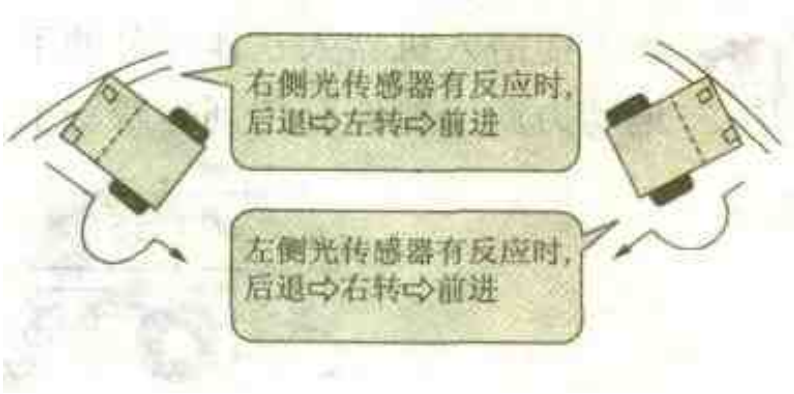
下面介绍"太郎"的动作。"太郎"在按下开始开关 5 秒后开始动作。其动作基本上只是前进。



那么，它走到摔跤场的边缘而不折回来，岂不是要自己输分。



"太郎"前面的左右都装有光传感器。无论哪只光传感器感知到摔跤场边的白线时，机器人就会开始拐弯向摔跤场的内侧前进。





这就是"太郎"的基本动作。只要有这个动作就不会自输，而且还能够继续与对手战斗。





是的。这个动作是相扑机器人所必需的。但是，仅靠这一基本动作，要正面去碰撞对手机器人，那是不行的。



所以，"太郎"上安装有超声波传感器。



是的。超声波传感器能利用声波的反射发现对手机器人。

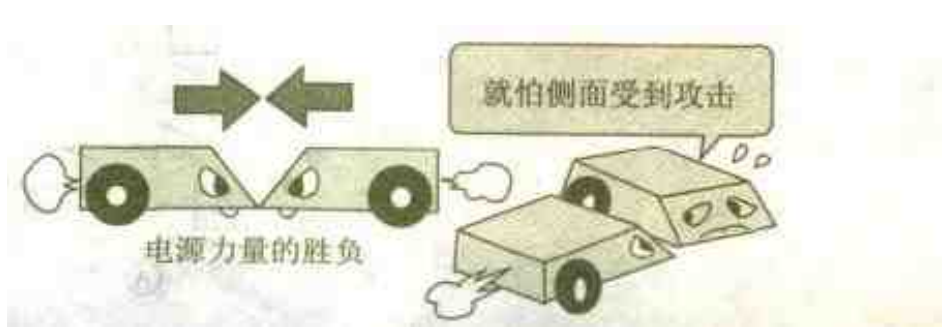
"太郎"上，左右各装有一只超声波传感器。



为什么不装在前面而装在左右呢？



传感器的安装位置有各种方案。用什么方案，得想想机器人大力



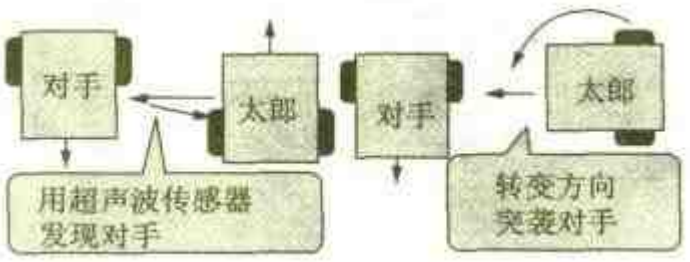
士的弱点在哪里？



从正面相碰时，电力决定胜负，而从侧面去顶时，我想对手就会显得弱。



是的。不敌侧面攻击是机器人大力士的弱点之一。所以，将超声波传感器安装在左右，当与对手机器人错位时，改变方向能从侧面碰撞对手的可能性就会增高。当然，这不过是一种方案。请大家按照自己的思路逐渐进行改善。

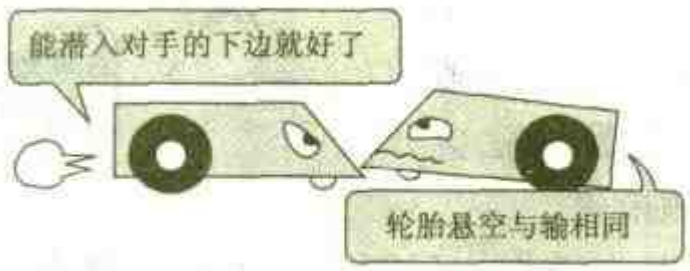




机器人大力士还能想到其他什么弱点吗？



如果能潜入机器人基体部分的下方，就能使对手悬空，这样对手的轮胎离地动力就不能充分传到摔跤场上，从面可以将其很简单地顶出场外。





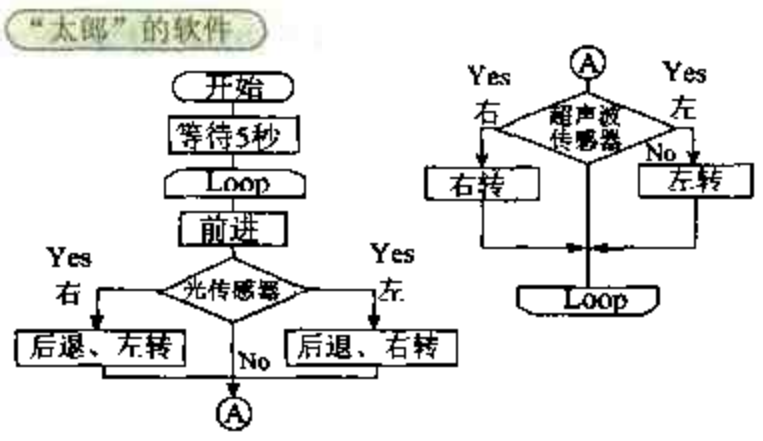
是的。所以为了不让对手潜入，关键是机器人与摔跤场之间的空隙要尽量狭小。可以考虑安装用电话卡制作的围档。



“太郎”的构造和动作概要已经明白了。太郎基本上是作前进动作，光传感器检测出摔跤场边框时改变方向向摔跤场的内侧移动，超声波传感器发现对手机器人后立即向对手前进。



是这样的。这些动作均由软件(程序)来控制。“太郎”的动作用流程图记述如下：



下面继续介绍至今已谈到的各个部分的详细原理及制作方法。

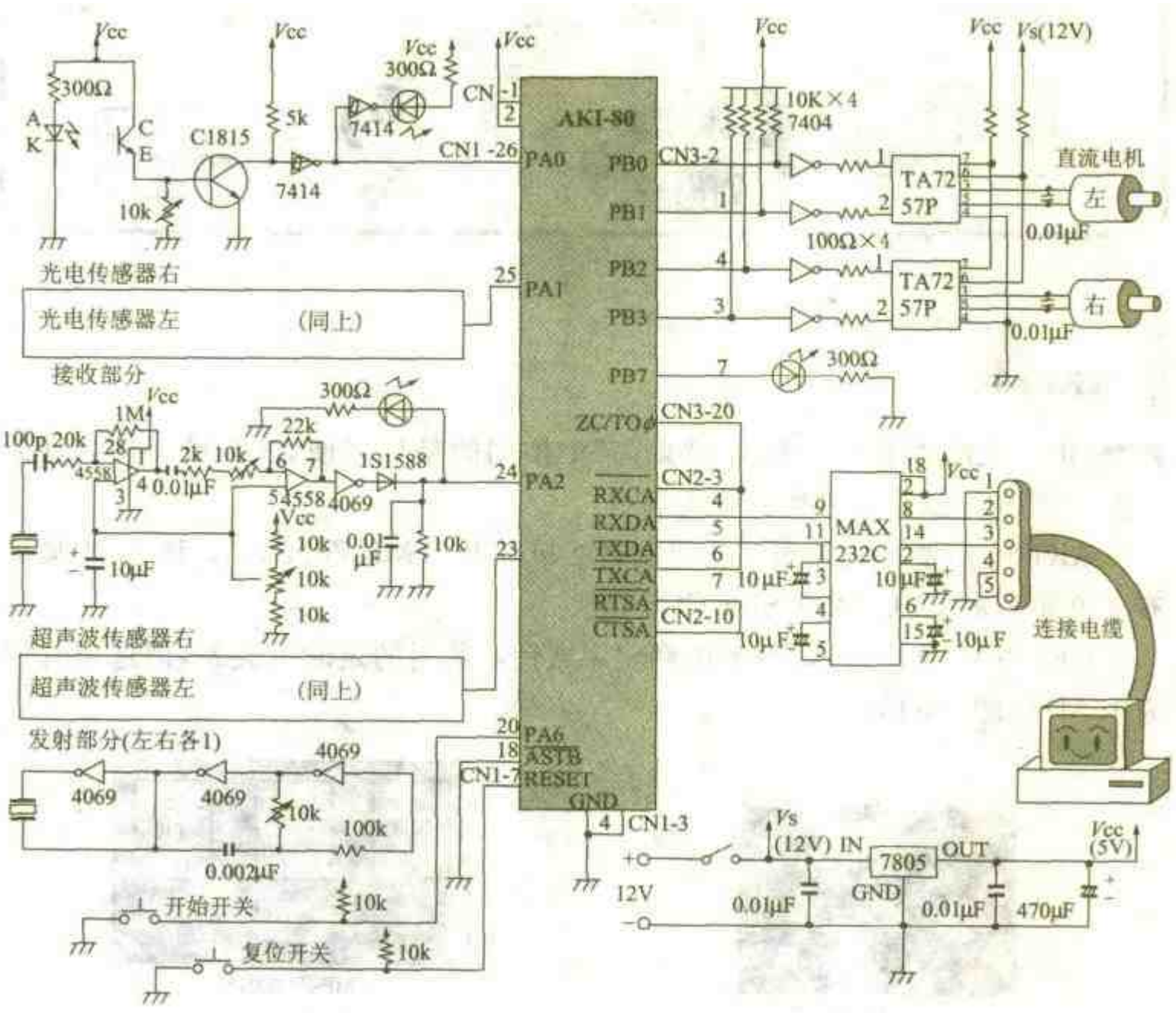
“太郎”的全电路图示在下边，请看一下。



已经知道“太郎”具备了自立型相扑机器人必要的基本功能。所以能够参加比赛大会。



但是，要在大会上胜利进军，还需要更多的好点子及技术。例如，最近许多很强的机器人都应用真空吸引技术，增大机器人与摔跤场间的摩擦力，这就需要增大机器人的重量。希望在“太郎”的基础上产生出更强大的机器人。



“太郎”的电子电路

练习题

1. 考虑相扑机器人最低限度所必需的功能。
2. 考虑相扑机器人的弱点。

（参照书中内容给出答案）

3 微机板的使用方法

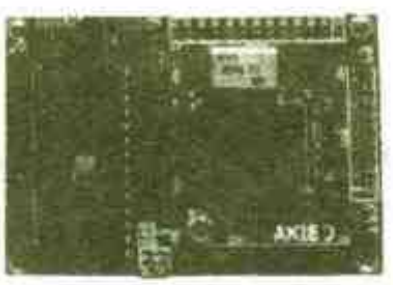


AKI-80

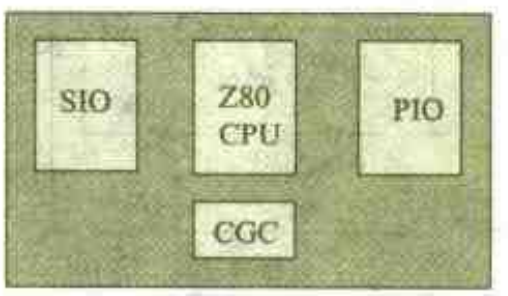
市面上出售多种微机板。“太郎”中使用的是价格便宜、小型而易处理的 AKI-80(秋月电子通商公司制造)。

AKI-80 中使用的是东芝公司制造的单片 IC TMP-Z84C015。该 IC 中集成有 Z80 系列的 CPU、PIO、SIO、CGC 等。

AKI-80 有几个版本,但考虑到使用通信功能时的定时等关系,请选用时钟为 10MHz 的 AKI-80。



AKI-80



TMP-Z84C015



将 AKI-80 的 CPU 视同 Z80 即可。



视为相同也没关系。还有,AKI-80 有成品也有配套零部件,选择便宜的元件组装亦好。组装时,要使用头细的陶瓷型电烙铁,在注意防止静电的同时要谨慎操作。

PIO(并行输入输出)



集成在单片 IC 中的 PIO 有 A、B 两个输入输出口。这些口到底用作输入还是用作输出,由程序来指定。“太郎”的情况,A 口作为输入用、B 口作为输出用。



总之,光传感器等输入数据时用 A 口,将数据输出给电机等时用 B 口。



这时,程序怎样写好呢?



为了 PIO 的使用,每个口都准备有数据和指令两个地址。



内部 I/O	通 道		地 址
PIO	A 口	数据	1C
		指令	1D
	B 口	数据	1E
		指令	1F

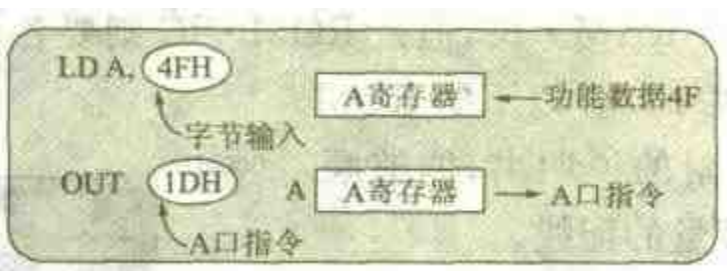
指令地址中,送入如何使用的数据,以决定口的任务。这种任务的指定由程序的最初部分描述。

功 能	数 据
字节输入	4F
字节输出	0F

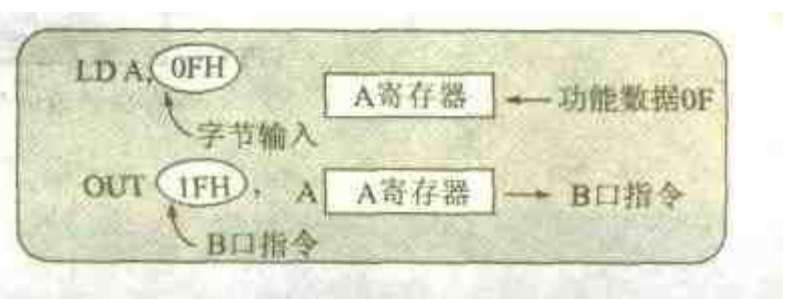
〈口功能的指定数据〉

(例)

指定 A 口为输入口

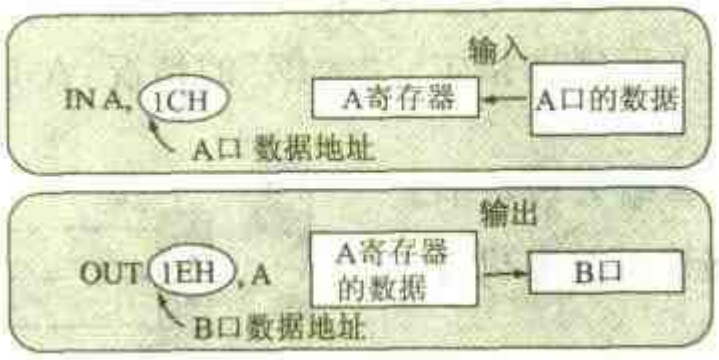


指定 B 口为输出口



若要从口输出输入数据时，要进行口地址和数据地址的指定。

（例）



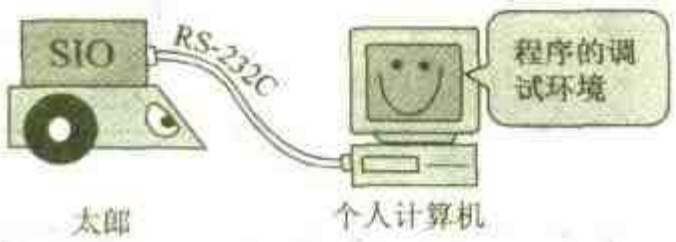


有一些难度。最初，作为例子照样使用就可以吧？

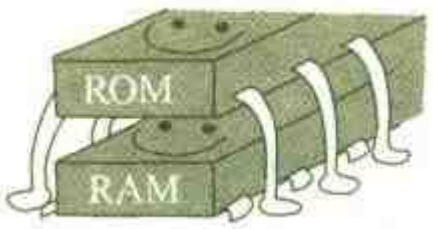
SIO（串行输入输出）



SIO 是在进行程序调试时，用于连接“太郎”和个人计算机。



存储器（RAM、ROM）



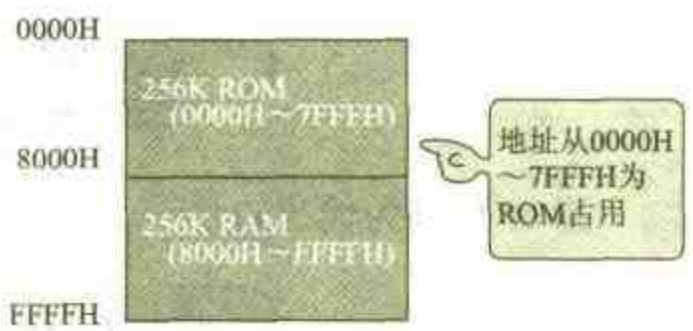


先来说明 AKI-80 的存储器。AKI-80 中，RAM 和 ROM 两者都带有。为小型化，RAM・IC 放置在 ROM・IC 的下方。因而，装上 ROM・IC 后看不见 RAM・IC。

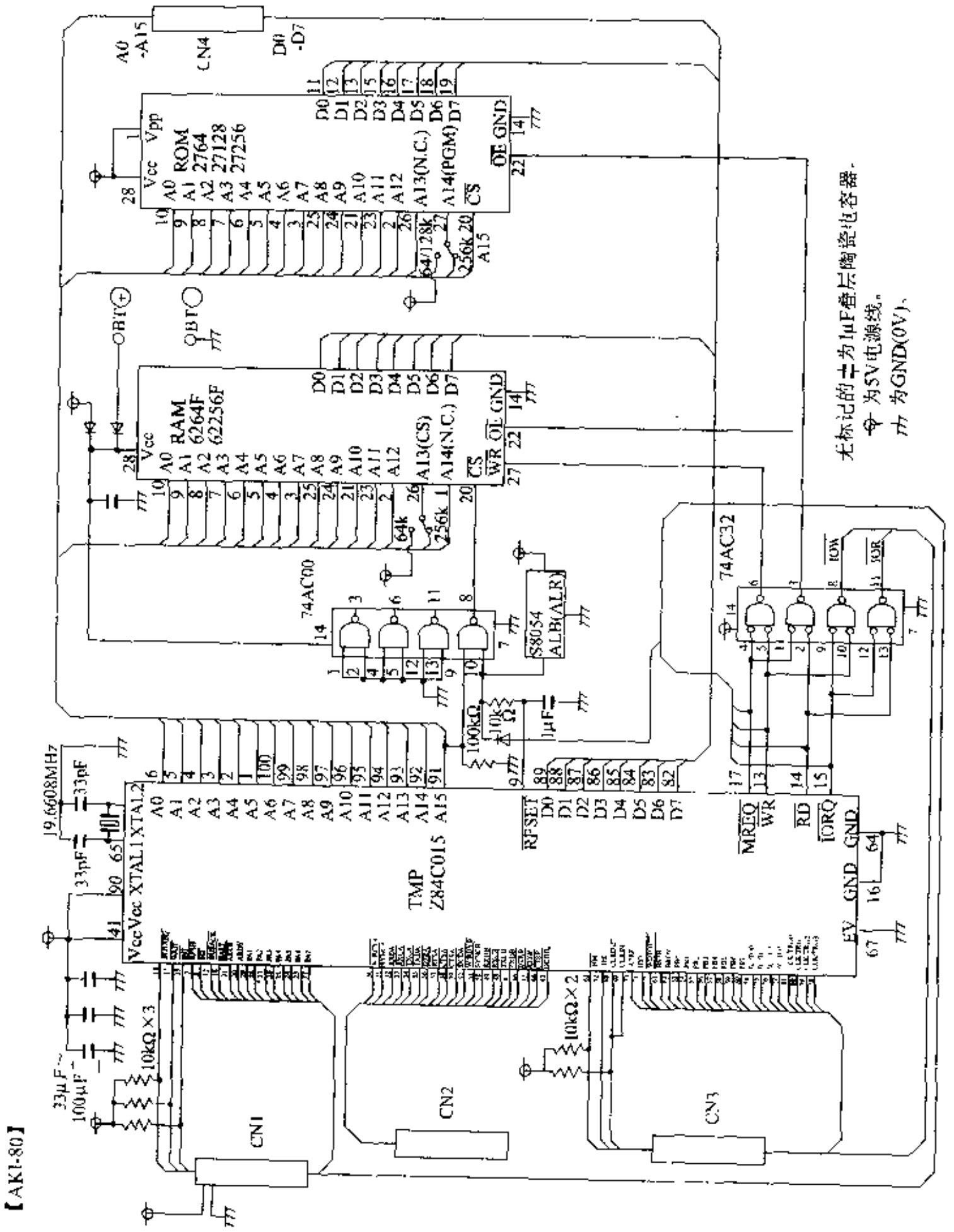
AKI-80 中包含着 RAM・IC，但是 ROM・IC 则要各自另外去买。需购入几个 256K 字节、速度 100ns 以下的 ROM。

按照 AKI-80 所附的说明书，能够顺利进行选择存储器容量的配线。

存储器 ROM 的地址分配为：0000H～7FFFH，RAM 为 8000H～0FFFFH。



<存储器地址分配图>



练习题

1. AKI-80 上使用的单片 IC、都集成有哪些功能？

2. 确认 PIO 的使用方法。

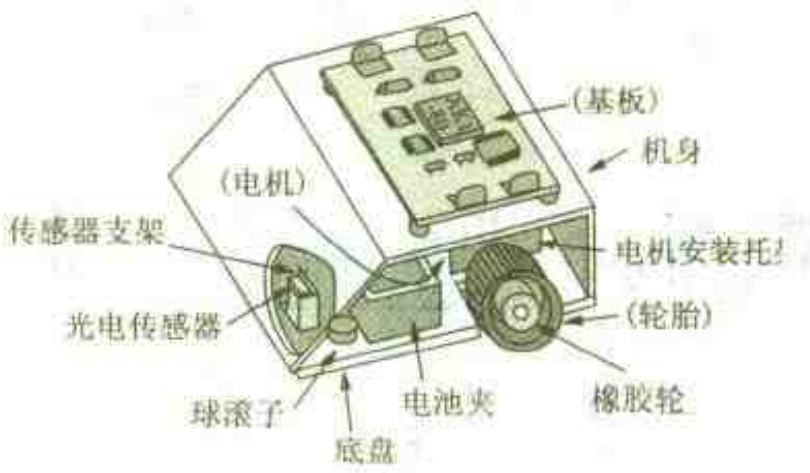
（参照书中内容给出答案）

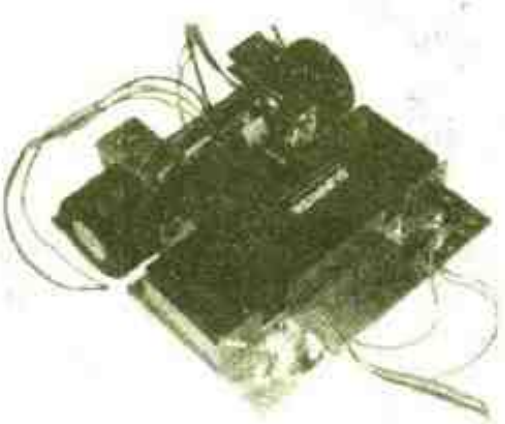
4 机械部分的制作方法



工作准备

先来看看“太郎”是都由哪些部分组成的。

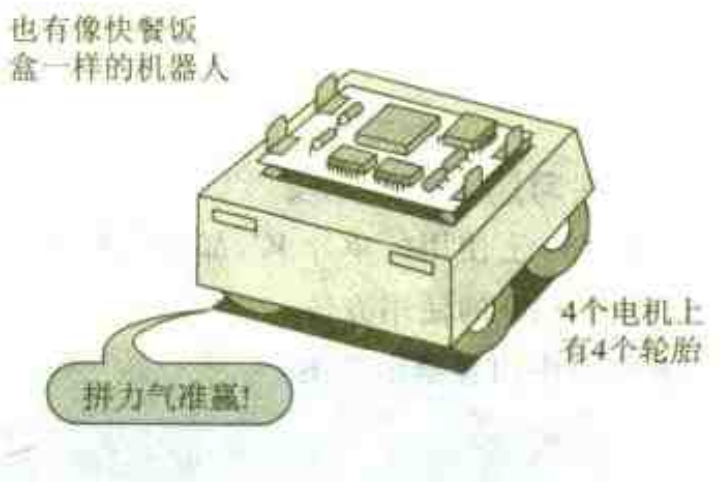




底盘和其上的零件

要做底盘、机身、橡胶轮、电机安装托架、电池夹、传感器支架等等。机身为楔形外形，其他的外形也可以吗？

按照“机器人相扑比赛大会”的规则，机器人的大小和重量有限制（长、宽 20cm 以下，高度无限制，重量 3000g 以内）。只要在限制的范围内什么外形都行。“太郎”是使用两个电机的两轮驱的楔动形，前轮用的是球滚子。当然



也有四轮驱动的以及做成快餐饭盒式的机器人。两轮驱动的是最一般的样子。


对于初学者，这种形式是最容易做的。

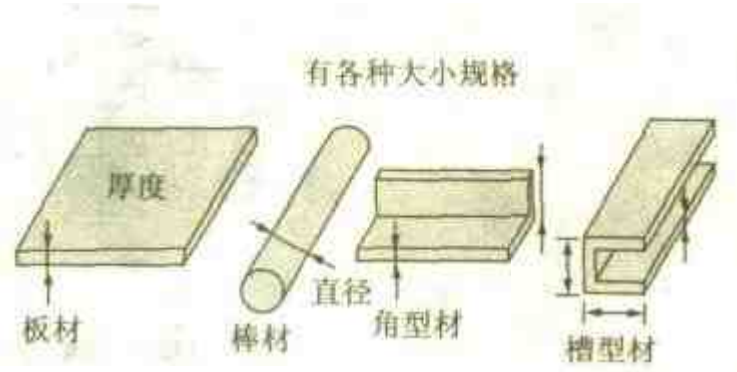

电机只有两个，留出空间好配置其他零件。


使用什么材料？


使用铝材。铝柔软易加工，同样强度下比较轻，因而很适于制作重量有限制的相扑机器人。


铝板和铝棒什么地方有售？


日杂店及经营机械工具的商店都能买到。他们备有各种厚度的板及各种直径的棒料。也有按要求切成所需大小的。而且角型材及槽型材也使用便利。

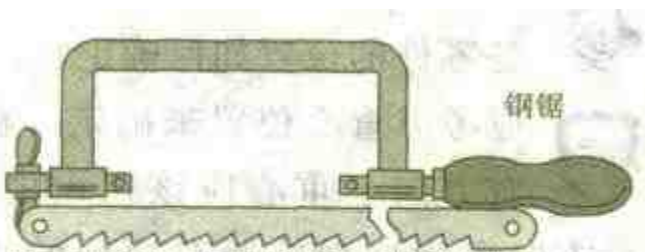



都用什么工具呢？

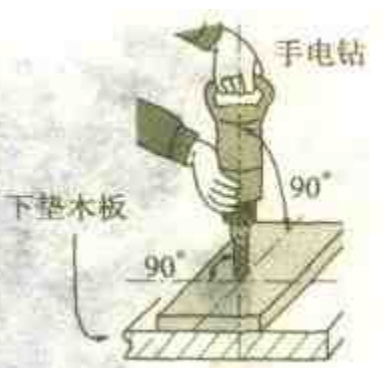

铝材要切割、要打孔、要弯曲。切断要用切割机或钢锯，打孔要用台式钻床或手电钻，弯曲则要用到折弯机或虎钳和锤头。


好像很难呀。


放心好啦。切割机、折弯机、台式钻床是什么地方都有的工具。用这些工具，制作会很简单。用钢锯、手电钻、虎钳也能制作。








只是切割、打孔、弯曲吗？

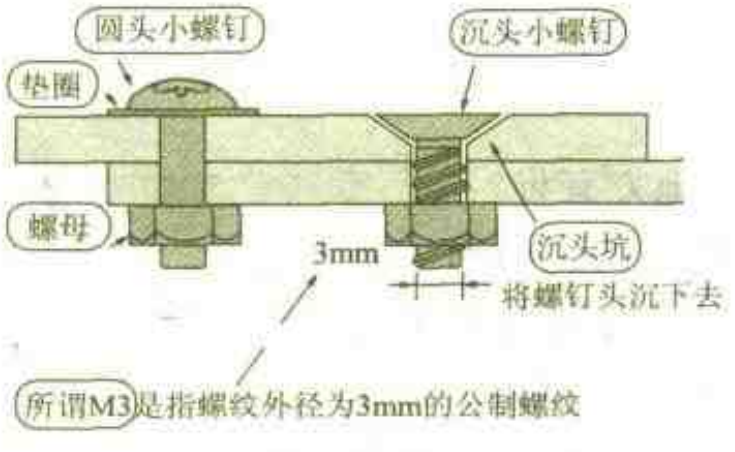

橡胶轮的制作中要用车床。也有用市售遥控汽车用的橡胶轮的。各种尺寸的都有出售。此外也有用丝锥作出螺纹的。


同一部件上的各零件用螺丝固定。

在日杂店中，螺丝、螺母、垫圈装成一小袋的各种尺寸的小螺丝都有销售。这里使用 M3（直径 3mm 的公制螺丝之意）的沉头小螺钉和 M2 的圆头小螺钉两种。

也有各种不同长度的吗？

同一直径的各种不同长度的小螺钉都有售，过长时可用卡丝钳剪断。

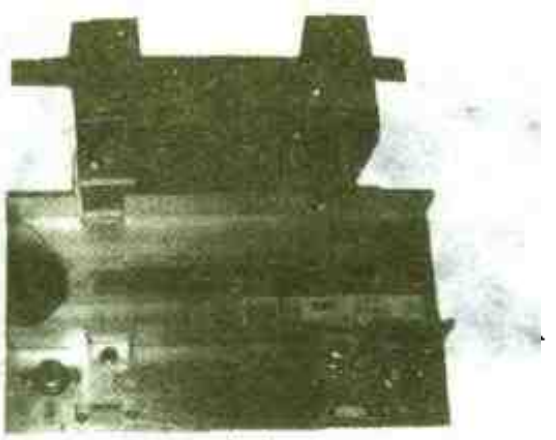


要用些什么材料和工具，已经明白了。

那么来看各个零件的设计和制作。参照 P157～P158 的设计图。

底　盘

底盘要承载许多零件。和对手相碰中如果变形，零件的位置就会错乱，所以要使用厚度 3mm 的结实的板材。



底盘

各零件的位置如何确定？

应考虑重心位置来确定。在左右方向上重心应该在正中、前后方向上为增强轮胎对地面的扒紧力要稍偏向驱动轮一边为好。移动电池的安放位置即可调节重心位置。

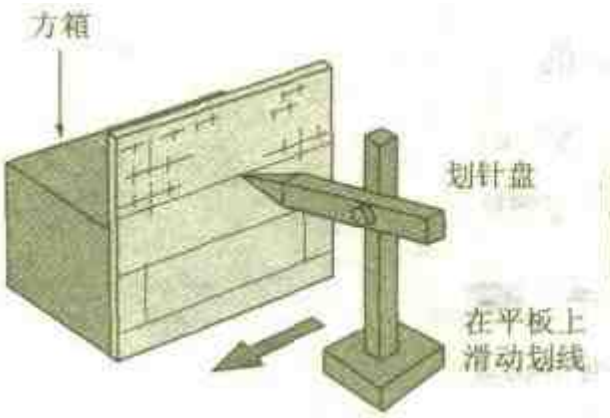
请给我们讲一下工作的顺序。

按照设计图在铝板上划线以便准确地切割，要在打孔的地方中心划个印。这叫“划线作业”。

底盘制作顺序

① 材料切成 185mm×190 mm，或请店里切料。
② 划线作业
③ 打孔并作沉头锥坑
④ 用钢锯切割
⑤ 用锉刀精加工
⑥ 打螺纹

若使用专用工具就能准确完成划线作业

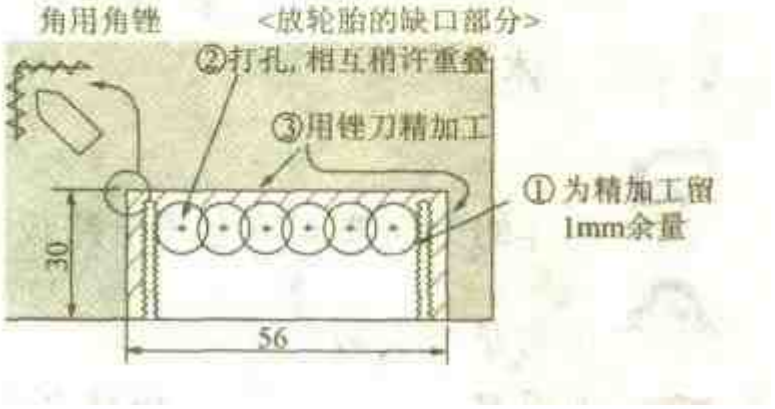




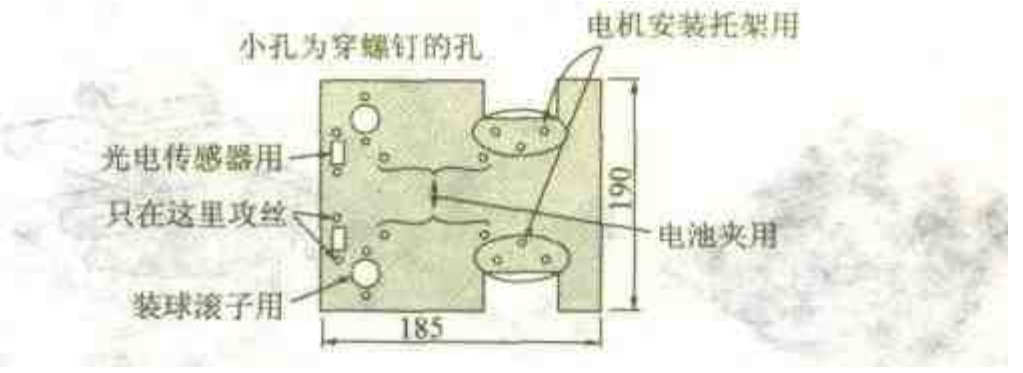

划线作业


工作中困难的地方在哪里？

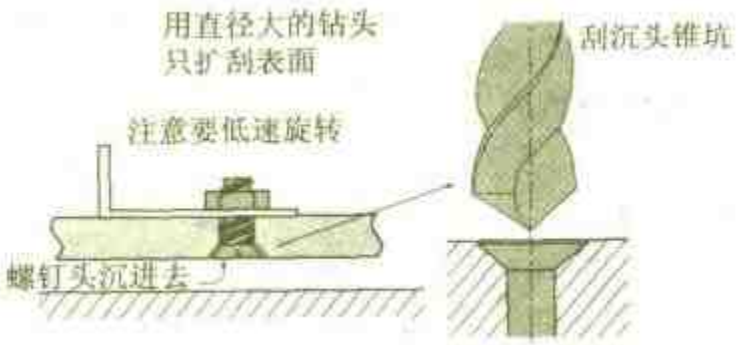

因为钢锯对安装轮胎的缺口部分及光电传感器用的孔不能加工，所以要使用钻子和锉刀。




为了将零件固定在底盘上，要打许多拧小螺钉的孔，这时要注意些什么？




打穿 M3 螺钉的孔用直径 3mm 的钻头，打穿 M2 螺钉的孔用直径 2mm 的钻头。由于钻头旋转振动，打好的孔直径约为 3.2mm 和 2.2mm，所以螺钉可以顺利穿过。小螺钉从外穿向内用螺母固定，麻烦的是底盘的底侧会露出小螺钉的头。为此，作沉头锥坑，将螺钉头沉进去。

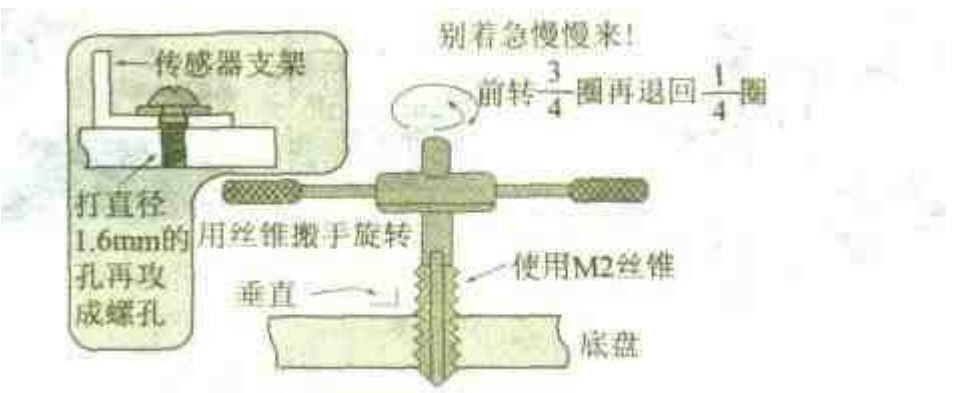




相扑机器人和摔跤场间的间隙很小，所以将螺钉头沉入底盘是很有必要的。



只是在固定传感器支架的地方要攻 M2 的螺纹。



基　体



基体的上面要安装电子电路板。前部为使对手悬空而折弯成楔形。



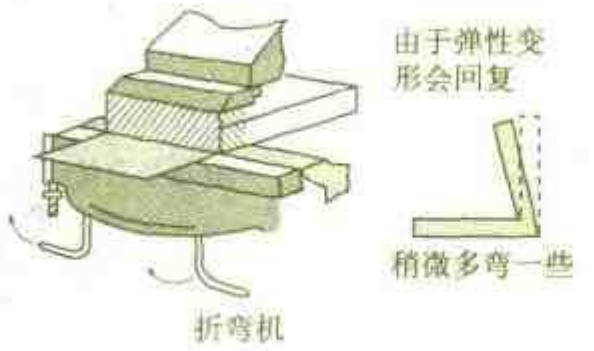
折弯是怎么做?



如果有折弯机，多厚的板材也能简单地折弯成各种角度。但是，力取掉后其变形会像弹簧那样多少有点回复(回弹)，所以应估计其回复量多弯一点。



基体(背后摄影图片)

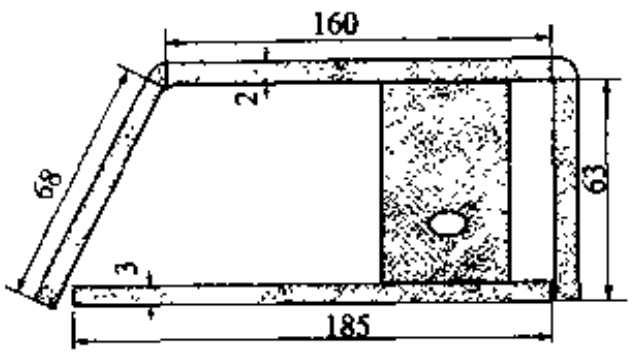




与做底盘一样，首先要做划线作业。



有一点要注意。折弯板时由于有一定厚度，内侧与外侧的尺寸是不同的。多数是线为内侧进行折弯，这样成形后外侧的尺寸约大一个板厚，请予以注意。

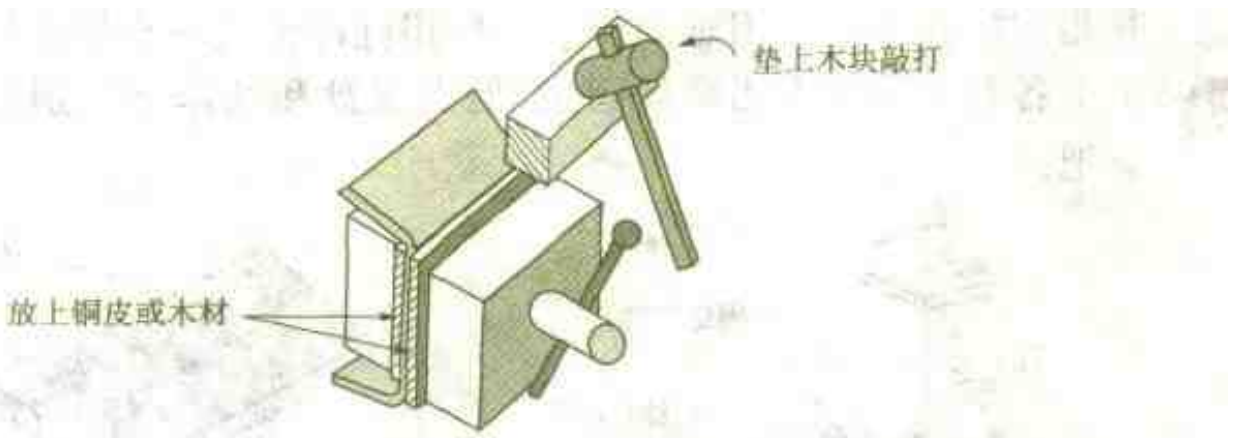




用虎钳也能折弯吗？



对 2mm 厚的铝材来说，完全可以。但要折弯得齐整应下功夫。为不伤材料要在虎钳与材料间夹垫柔软的铜皮或木材，锄头不直接敲铝材而是通过木块敲打。而且，用划针等划出折弯部分的外侧刻上印即可完成齐整的折弯。



电机安装托架



电机安装托架，有连接底盘和机身、固定电机的作用。



电机的安装位置由车高决定。



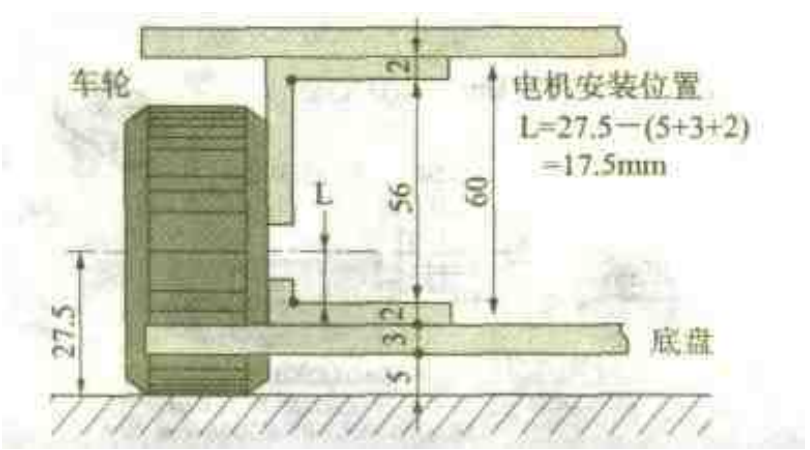
由于使用球滚子，前部的车高约为 5mm，所以后部也同样要设计为 5mm。



正确的钻开电机轴的穿孔是很重要的。

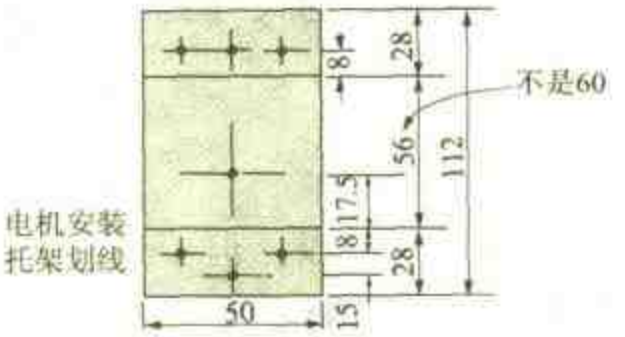


轮胎的直径为 55mm，所以电机的中心位置按下图确定。





考虑到板厚，应按下图划线：





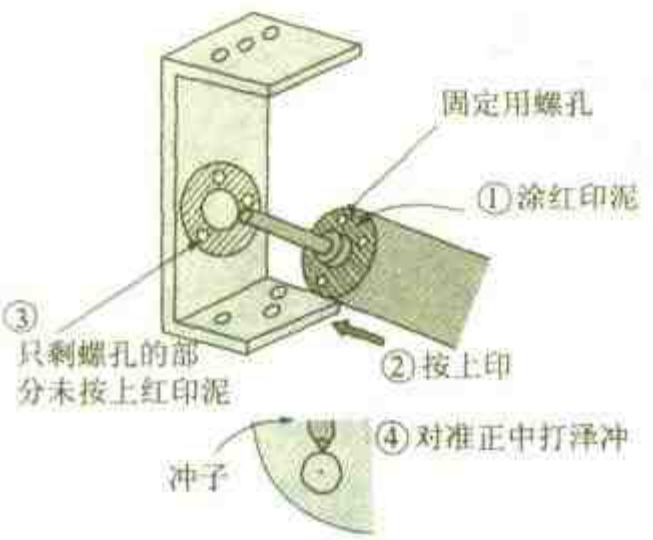
电机怎样固定在托架上？

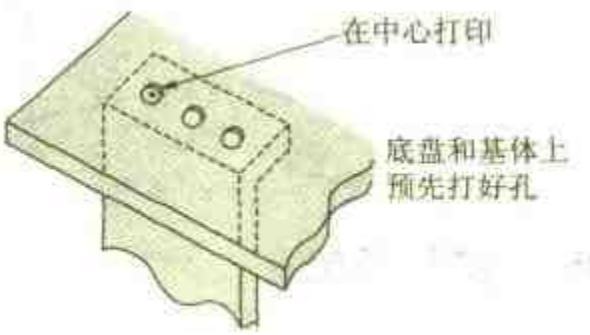


这次使用的电机上有预先做好的固定用的 3 个螺孔。但要在托架上钻与其相配的孔是较为困难的作业。使用红印泥是一个好方法。



上下各有 3 个地方用螺丝固定，但在底盘和基体上分别正确地钻孔我没有把握。







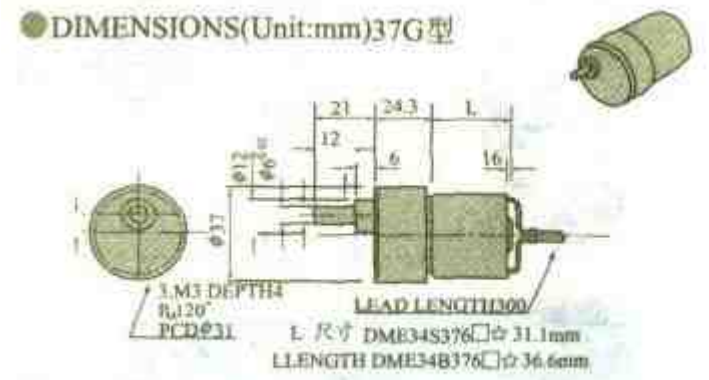
底盘和基体上的孔正好对上组装简单。



使用哪种电机好？



选择电机时要考虑形状、额定转速和扭矩等。这次使用的是与减速齿轮做成一体的传动电机，型号为 DME34B37G18A（日本伺服电机株式会社制）的直流电机。




如果扭矩大转速高，机器人会很强大吗？


很遗憾，扭矩与转速是成反比的。要产生大的扭矩速度就会下降，而想要产生高速度扭矩则会减小。和汽车的低速齿轮与高速齿轮的关系一样。


扭矩大的是可以强力顶对手的“顶推相扑型”，转速高的则为“速度相扑型”。


速度太快时，光电传感器感知到摔跤场边缘后，来不及转换方向就会飞出而“自杀”。还有，即使扭矩很大，但如果轮胎在摔跤场上扒不紧时还会产生空转打滑。所以既要使轮胎表面不易打滑，又要增大加在轮胎上的重量。


速度与扭矩过大是否无用？


使用大电机的确有利，但是有一个限度。



轮胎转一圈只前进($2\pi r$[m])，
所以转速为n(r/min)时，车速=$2\pi r \times n$(m/min)

车轮

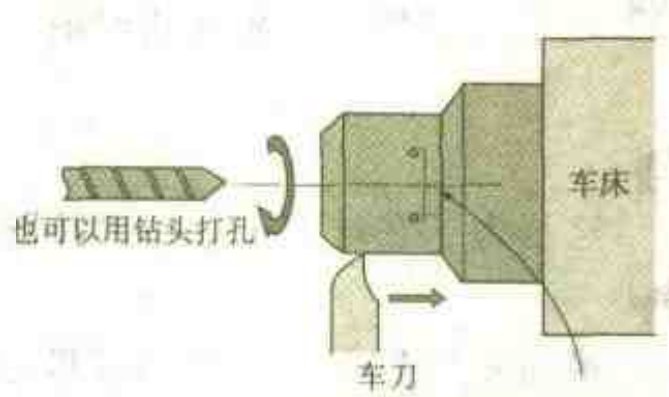

因为容易找到适当大小的遥控汽车用的海绵状轮胎，所以可以亲自制作与其相配的轮子。这可以简单地用车削制作。而场上也有将遥控汽车用的轮胎与轮子一起配对出售的。


怎么将轮子与电机轴固定？


钻两个螺孔，用顶丝固定。还要考虑海绵状轮胎与轮子间的防滑问题。


可以使用双面胶带纸。


车轮及其安装



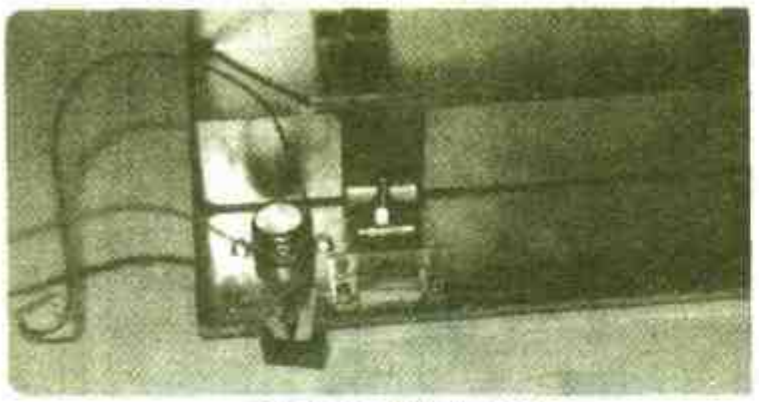
光电传感器、电池、电路板的安装



光电传感器支架、电池夹子等要用角型材加工自制。制作时要符合所用的光电传感器与电池的形状。电子电路板用四根六角柱头螺钉固定于机身上。



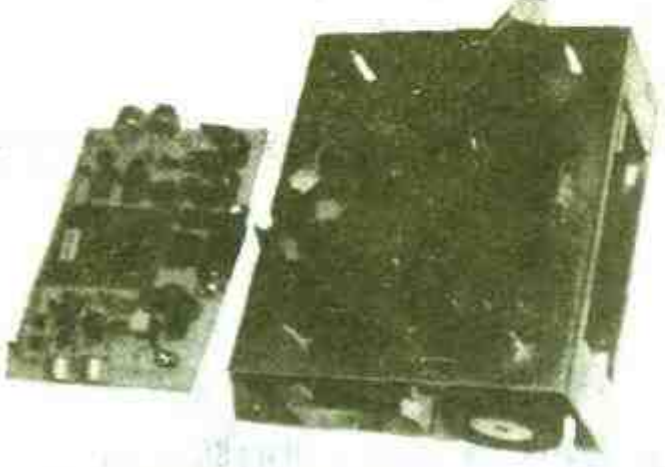
孔位置可与实物配作。



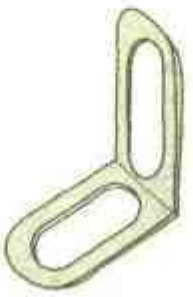
光电传感器的安装



电池夹子(加工角型材制作)



电子电路板的安装



L形架



六角柱头螺钉



小型球滚子(球轴承)



光电传感器的位置在边缘啊。



这是为了在球滚子未掉离摔跤场前检测摔跤场边缘转换方向。



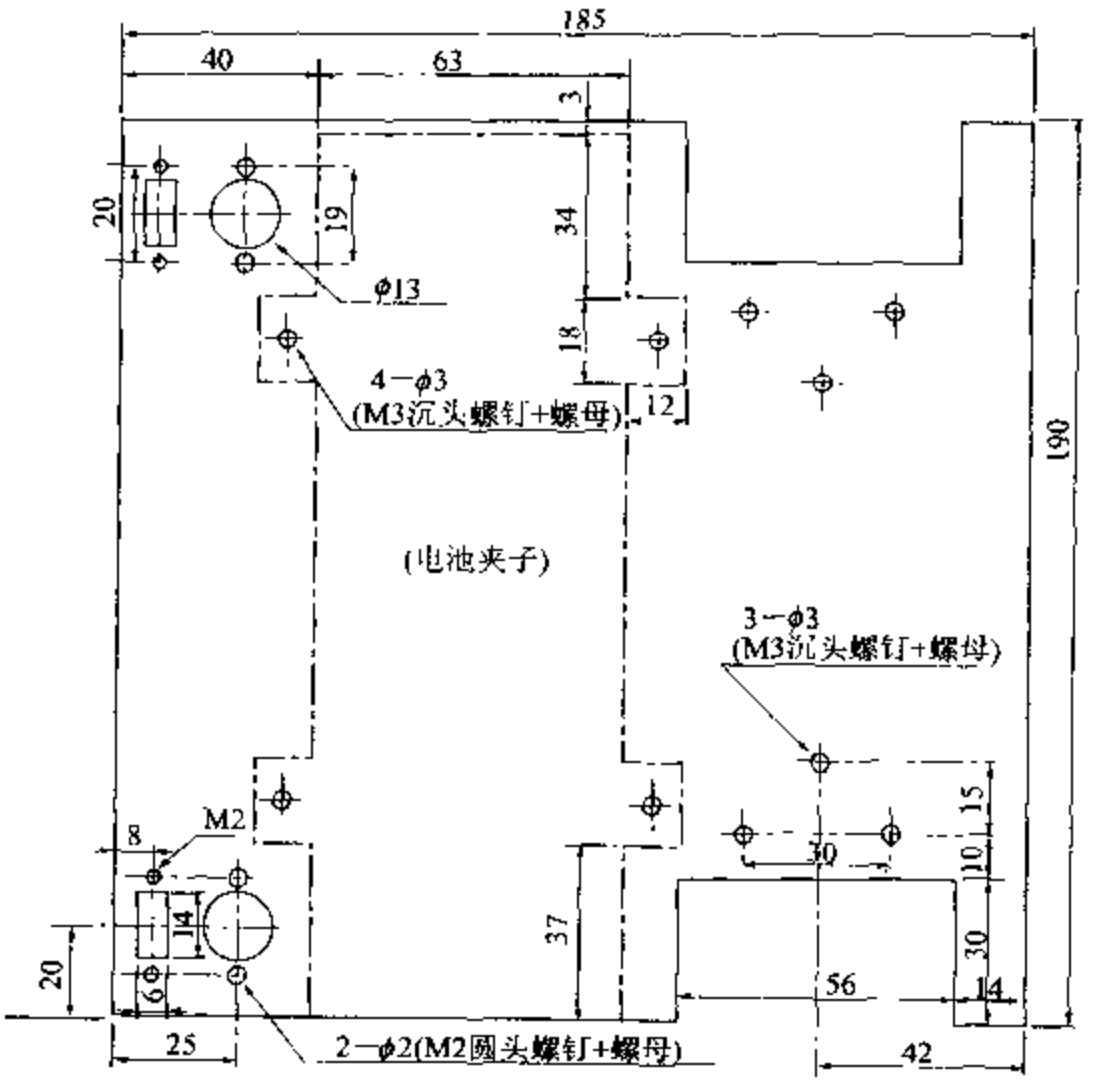
准备零件后用小螺钉和螺母进行装配。



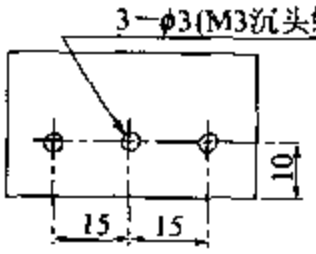
在手难进去的狭窄地方,将螺母粘在食指上进行工作会很简单。



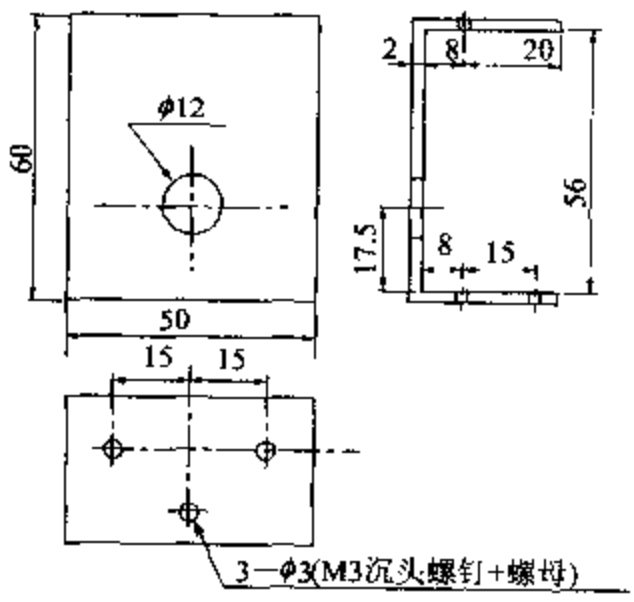
最后,将传感器、电机及电池的引线连接到电子电路板上。



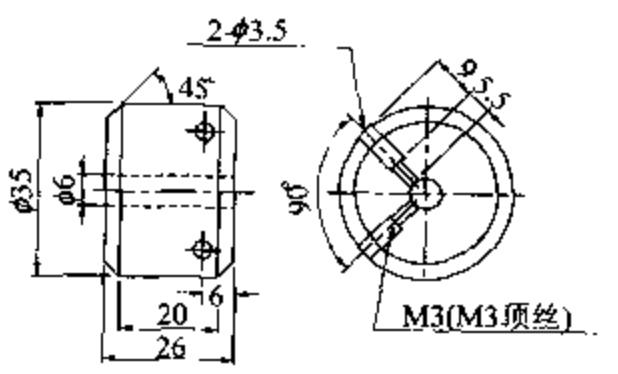
底盘设计图
材料
(190mm×185mm×3mm)
(注)3—φ3(M3沉头螺钉+螺母)是表示用直径3mm的钻头在3个地方打孔,再穿入M3的沉头螺钉用螺母去固定零件

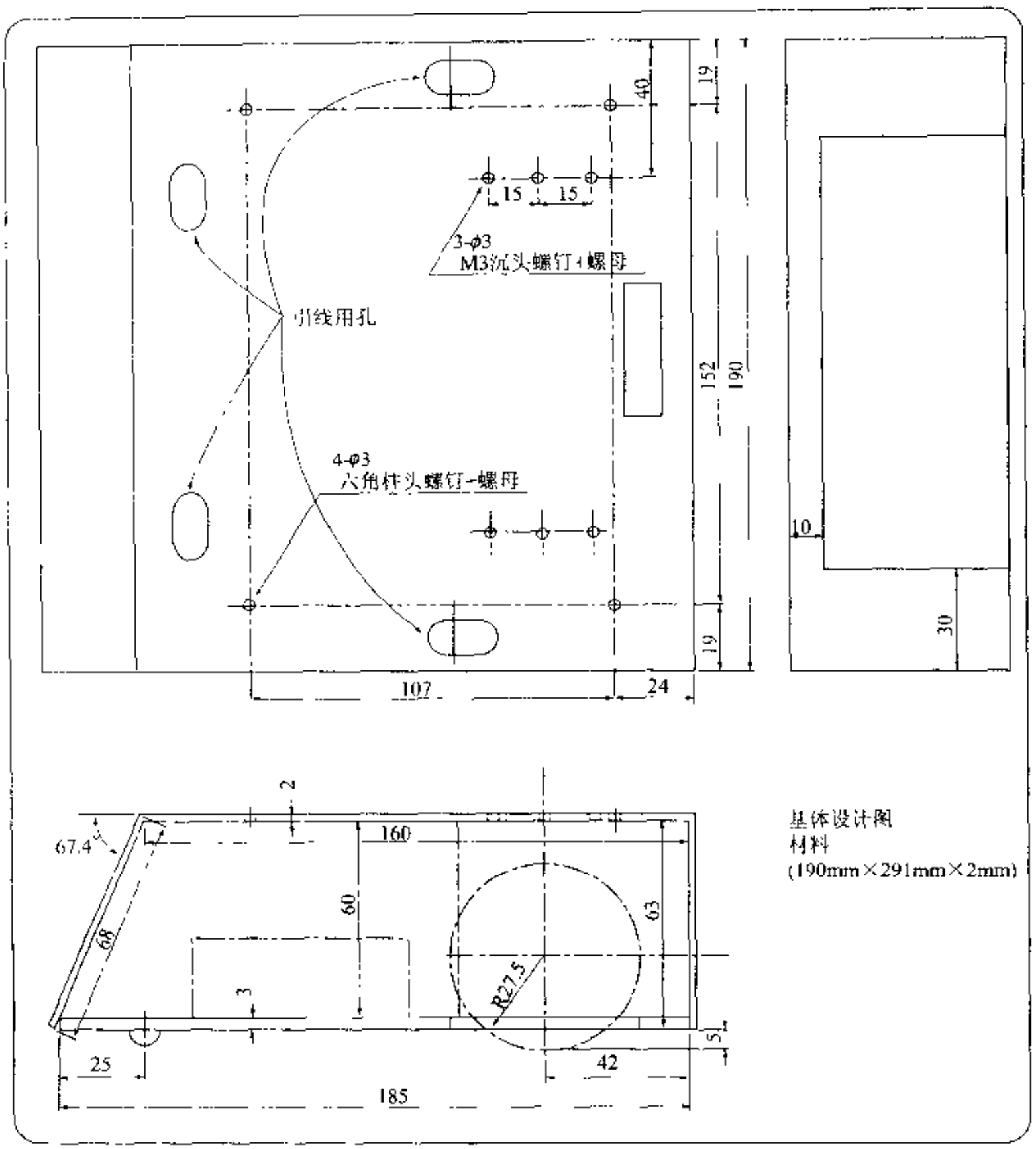


电机安装托架设计图
材料
(112mm×50mm×2mm)



车轮设计图





基体设计图
材料
(190mm×291mm×2mm)

为使引线能经受激烈的相扑角斗，应在机身的适当位置钻孔使其穿过，以使引线尽可能地短。

练习题

1. 举例说明钣金加工的一般工艺顺序及所使用的机械与工具

答案(参照书中内容)

2. 相扑机器人的机械加工中最需注意的地方在哪里？

解答

折弯板时要考虑到板的厚度。还要特别注意电机安装托架的加工。

5 电子电路的制作方法



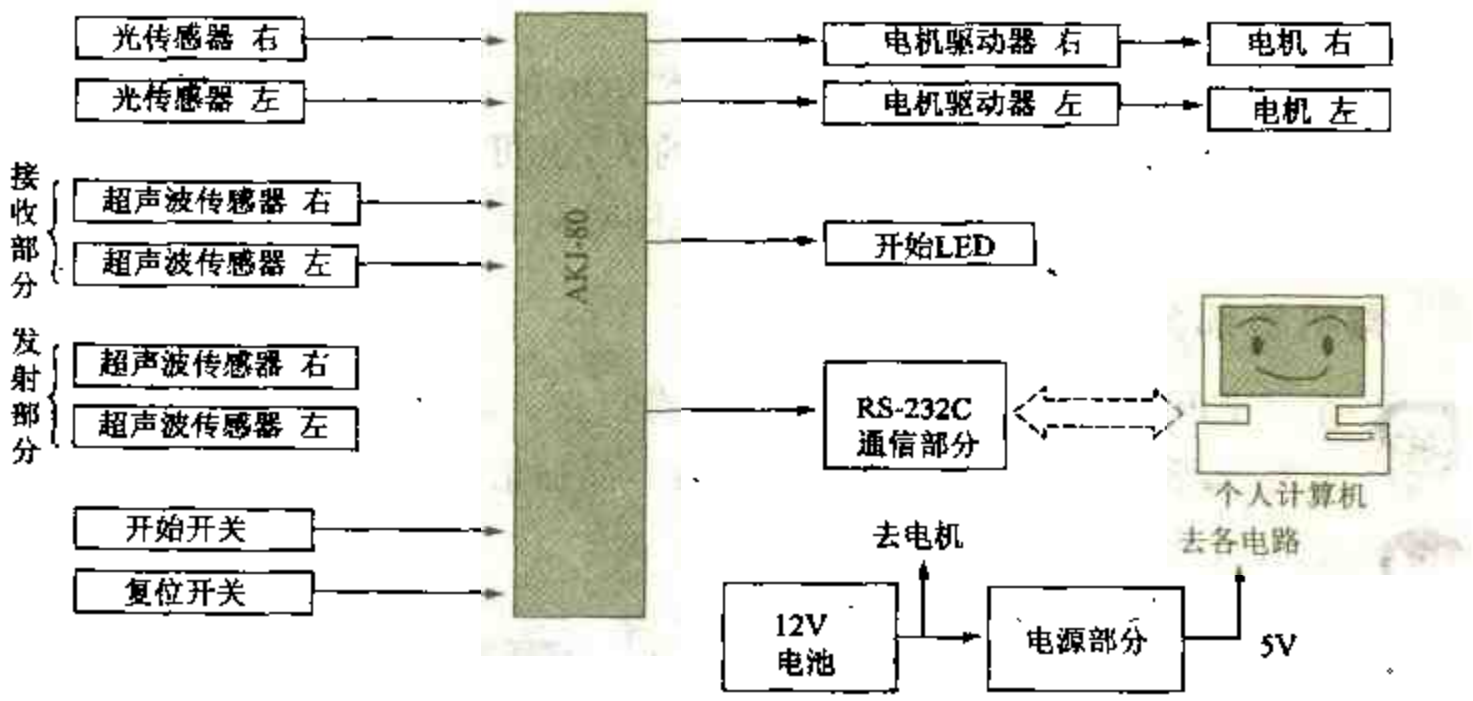
“太郎”的电子电路



在前面(P140)已经给出了“太郎”的电子电路图。这里示出其方框图。



该方框图以 Z80 微机板为中心，由光电传感器部分、超声波传感器部分、电机驱动部分、RS-232C 通信部分及电源部分等组成。



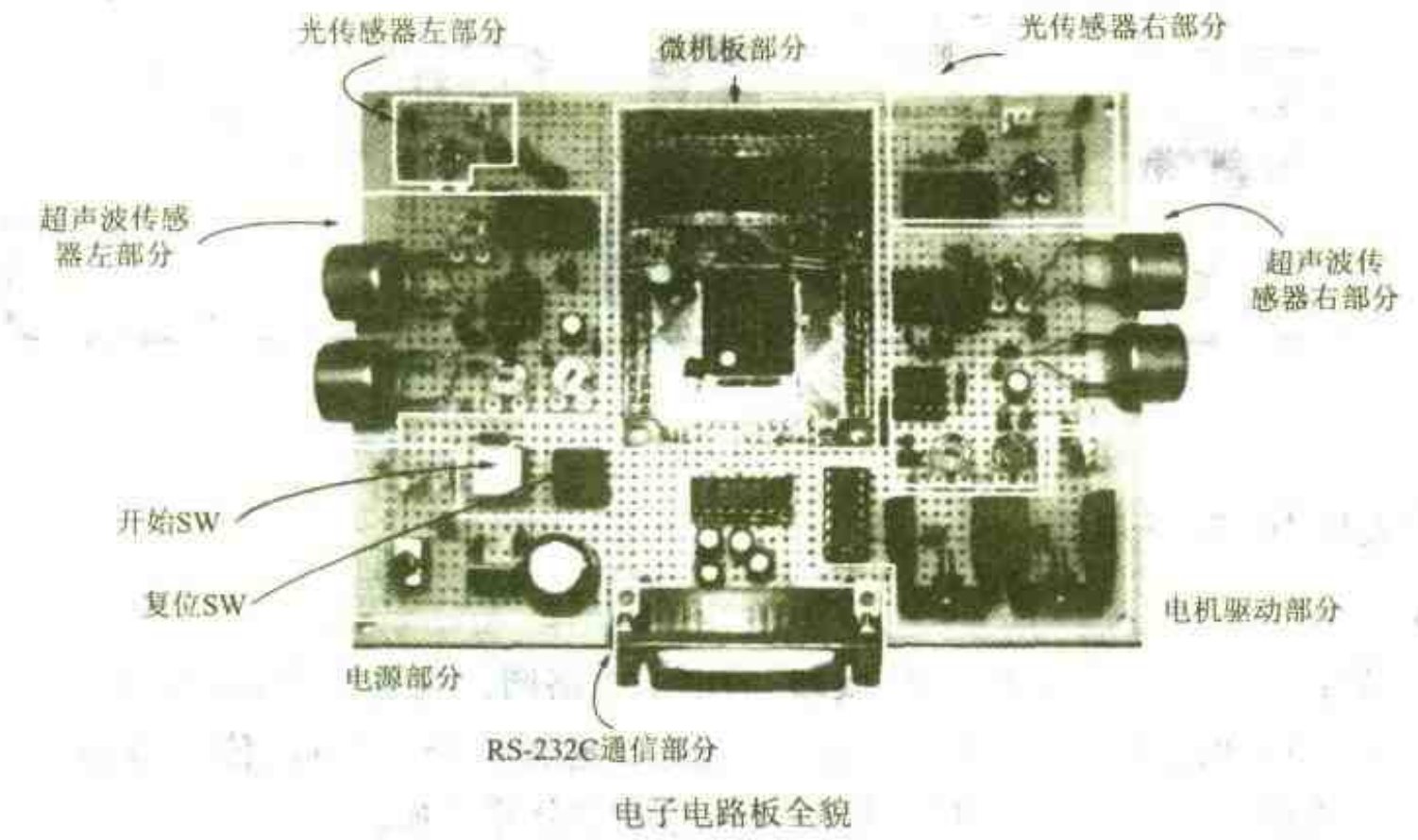
太郎的方框图



AKI-80微机板上，准备两个输入输出口。

A口和B口。

这些口如何使用需在程序中指定。“太郎”中分配A口为输入用、B口用作输出。



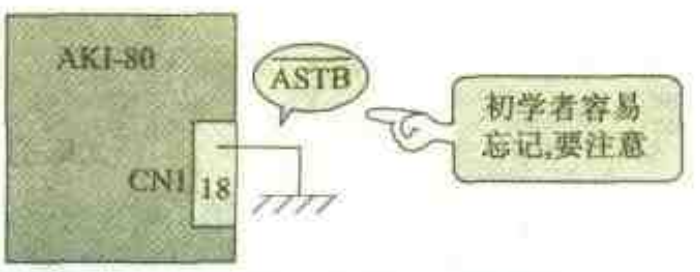
电子电路板全貌

将电子电路组装在一块11.5cm×16.0cm的通用印制电路板上。引线使用的是0.5mm的单芯导线。但有把握的人，也可以自己用蚀刻法制作电路。这里给出了电子电路板的照片，请与方框图对照。

微机板部分

在电子电路中核心部分是AKI 80微机板。因为有一些细的引线，请先仔细阅读所附的说明书后细心地组装。晶振请使用19MHz的。

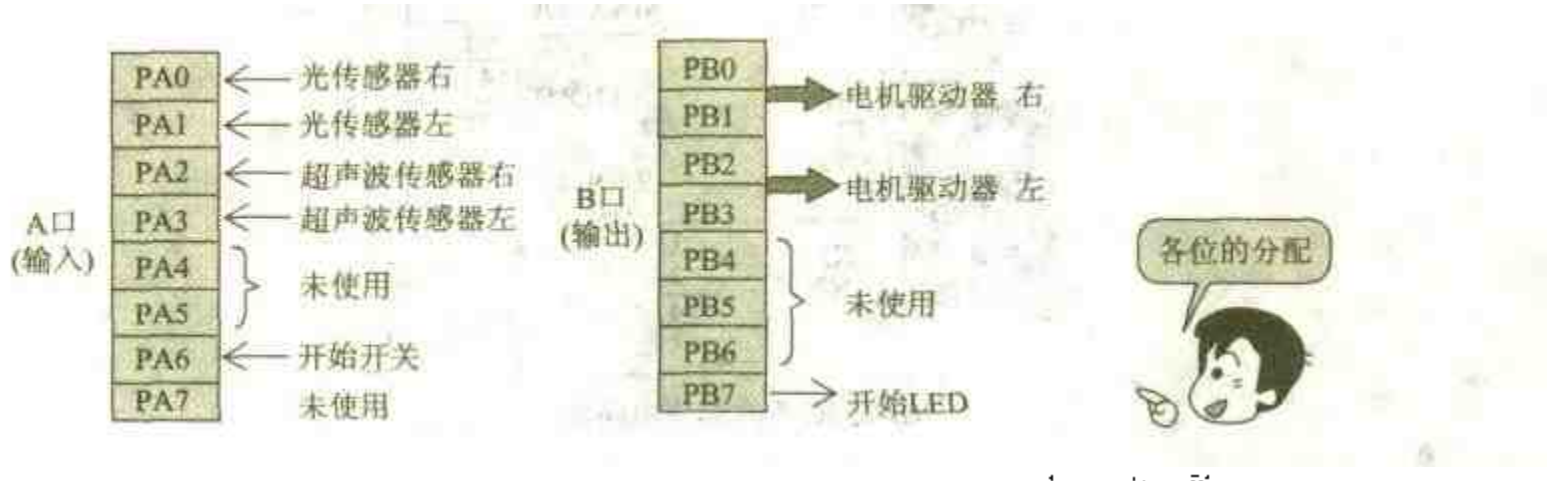
还有其他要注意的事项吗？



还有一个很重要的注意事项。如前所述，“太郎”是将A口作为输入使用的。为此要将微机板的$\overline{\text{ASTB}}$端(CN1-18)接地，否则会以很高的概率出现误动

作。

先决定输入输出口各位的分配。



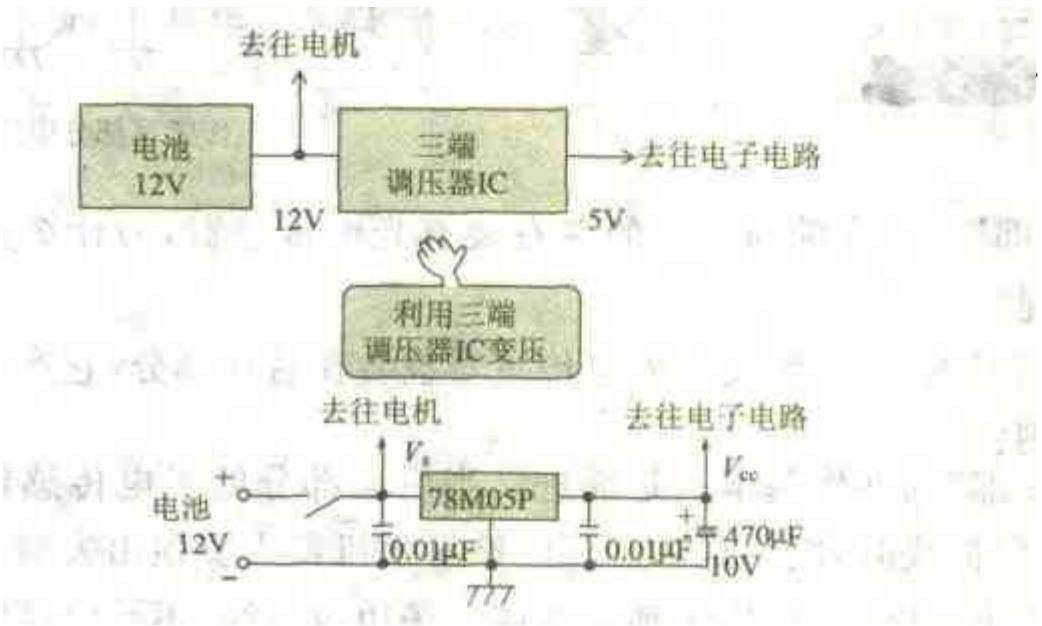
电源部分


"太郎"的电机要用 12V 电源，电子电路要用 5V 的电源。


就是说要使有两块电池是吗？


不是的。"太郎"不是使用两块电池，而是从一块 12V 电池，通过三端调压器 IC，取得 5V 电压。

"太郎"使用的电池是 12V 的充电电池。市面上出售有各种类的充电电池，可考虑规格、大小及重量等进行合适的选择。

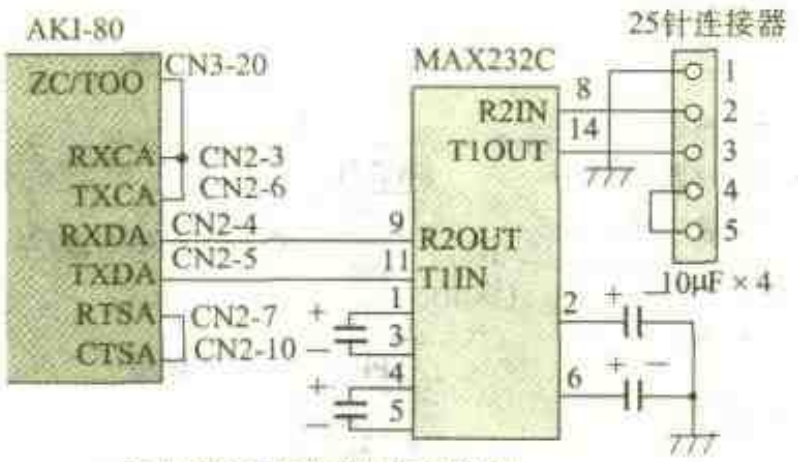


电源部分电路图

RS-232C 通信部分


RS-232C 是进行程序调试时，为将微机板与个人计算机连接起来而设置的电路。

电路板上安装 RS-232C 用的 25 针孔连接器。



RS-232C通信部分电路图

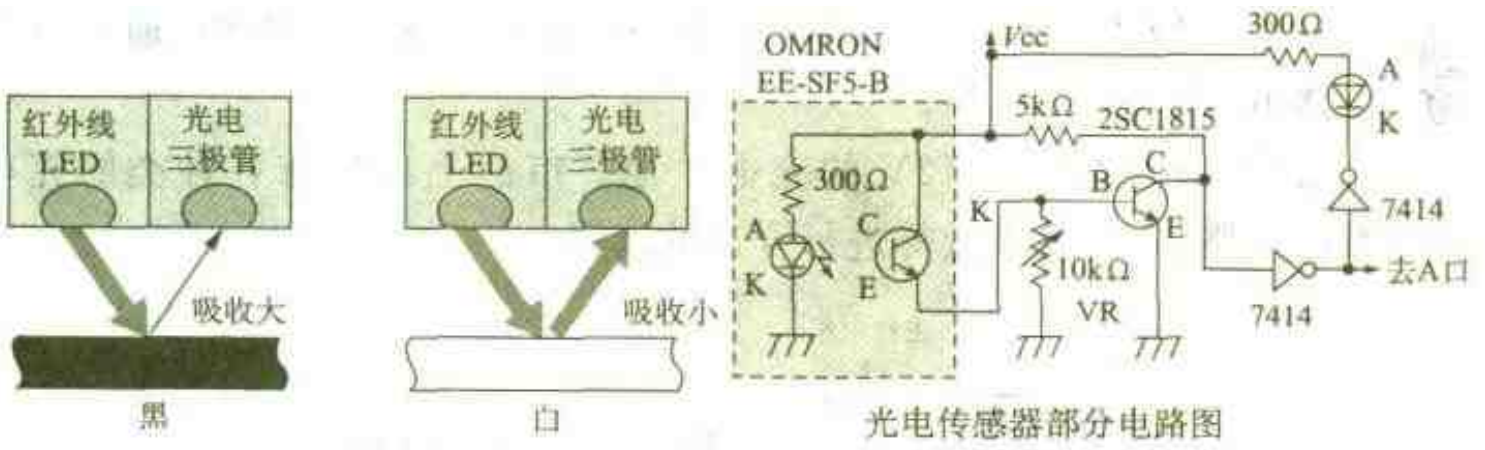
光电传感器部分



光电传感器的作用是检测摔跤场的白色边缘线是不是?



是的。它是利用黑白色对光的吸收率不同,使用的是反射型红外线传感器。为调整灵敏度时的便利,还装有 LED 监测器。



光电传感器部分电路图



“太郎”只是在前面部分的左右装有光电传感器,为什么在后面的部分不装呢?



你们考虑一下,假设将光电传感器也装在后面部分,它产生反应是在什么时刻?



“太郎”的动作基本上是前进。当后面部分的光电传感器检测到摔跤场的白框线时,岂不是“太郎”已被对手机器人要顶出去啦。



要顶出去时,即使检测到一些摔跤场边缘白线,不是已经没有有效的对付办法了吗?





“太郎”正是出于这种考虑而不在后面部分安装光电传感器。还有，如果给光(红外线)加以调制，还可提高可靠性、防止场地照明等引起的误动作。但是实验结果表明，这里示出的未调制的电路也非常实用。为便于区别摔跤场内的黑色与摔跤场边缘的白色，可观察着 LED 监测器进行 10kΩ 的半可变电阻器的调整。

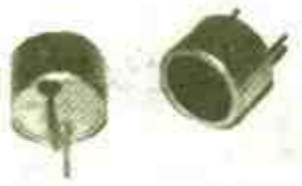
超声波传感器部分



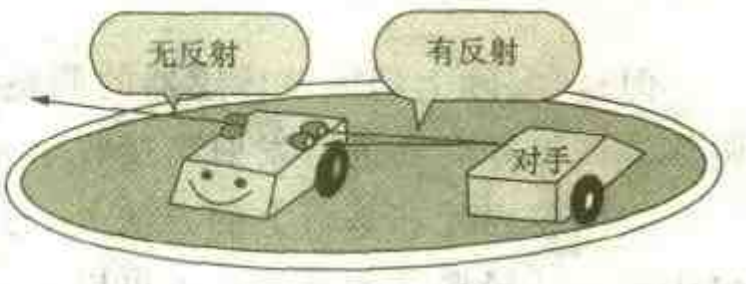
为发现敌人而设置的电路就是超声波传感器部分。



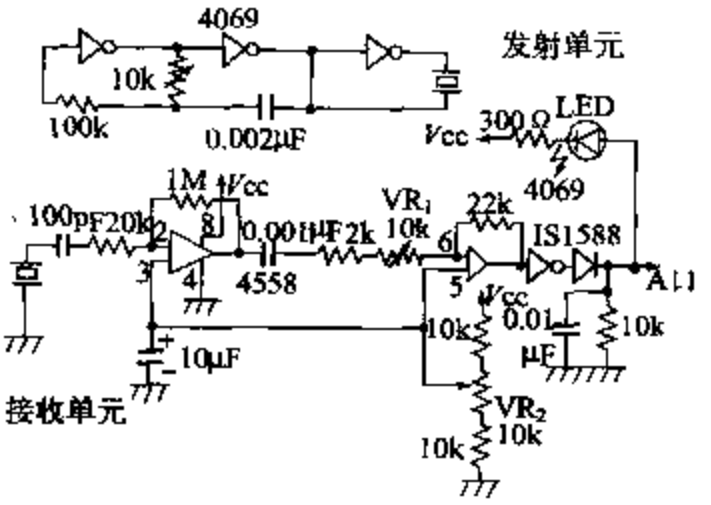
此超声波传感器是利用 40kHz 的超声波，根据其反射的有无来发现对手机器人。



超声波传感器单元



电路由发射部分和接收部分组成。虽然发射部分也可以共用，但“太郎”上携带有左右各自独立的发射部分和接收部分。也安装有调整用的 LED 监测器。



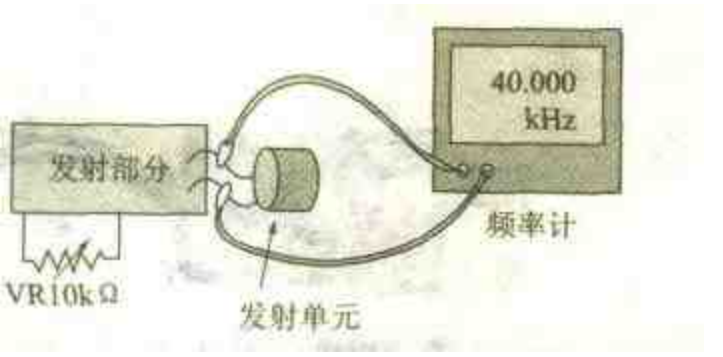
超声波传感器部分电路图



可变电阻有许多，怎么调整？



先调整发射部分。使用示波器或频率计，调整可变电阻 VR，使得加于发射单元的信号频率为 40kHz。



接收部分调整两个可变电阻。为的是边改变发射与接收单元间的距离、角

度、高度等，边感知处于适当位置的对手机器人。


没有确定的调整方法吗？


当然有，从电路的原理考虑有调整的顺序。

接收部分的调整

① 切断发射部分的电源；
② 将可变电阻 VR_1 调至最大；
③ 将可变电阻 VR_2 调至监测 LED 刚亮；
④ 降低可变电阻 VR_1，调至监测 LED 刚亮。

但是，实际上超声波传感器的调整很难按道理进行。建议反复耐心地进行调整。调整好后要进行发射与接收单元的防振处理。

除用胶固死在基板上的方法外，与其相反，也可以考虑在单元的下边放海绵垫等来减轻振动的方法。也可以用实验试试各自的想法。

超声波传感器的调整虽然麻烦，但一旦调整好，其后即能稳定地工作。

电机驱动部分

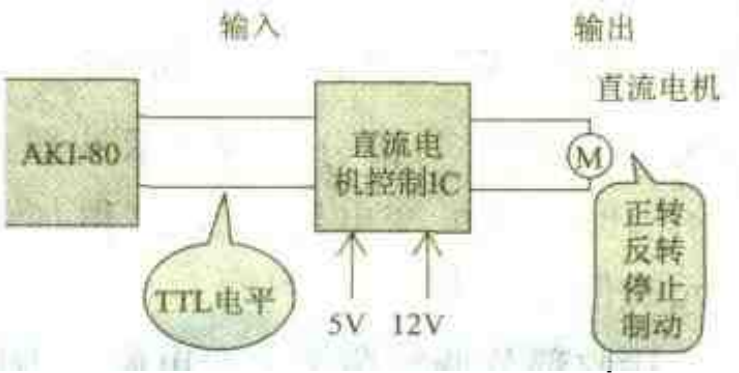

“太郎”使用两只直流电机。


微机控制中经常使用脉冲电机(步进电机)，为什么这里采用直流电机？


与脉冲电机相比，直流电机能够取得很高的旋转速度。如果将齿轮头装在直流电机上减速，还能获得适当的速度和扭矩。低速、低扭矩的脉冲电机，并不适于作相扑机器人的腿。“太郎”用的是日本伺服电机株式会社制造的型号为 DME34B37G18A 的转速为 277r/min 的电机和齿轮头。

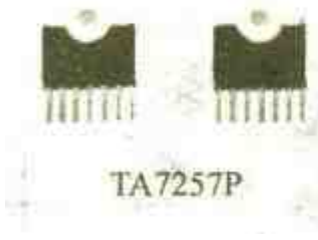

如何控制电机的正转、反转与停止？电机旋转时也要控制很大的电流吧？


直流电机+齿轮头



直流电机可使用大电流用的三极管或 FET(场效应管)进行控制。而且市面上也有直流电机控制用的便利的 IC 出售。“太郎”就采用这种 IC。这里来介绍“太郎”使用的东芝公司制直流电机控制用的 IC-TA7257P。

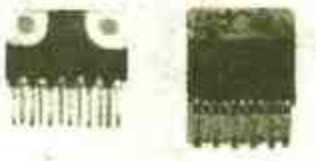
该 IC 按照输入到每个输入针上的 TTL 电平,可选择电机的 4 种动作模式。


TA7257P

输入	输出	模式
1　1	L　L	制动
0　1	L　H	正转
1　0	H　L	反转
0　0	高阻抗	停止



IC 能够控制的平均输出电流为 1.5A。因为“太郎”使用的电机的规格为 12V、0.76A,所以即使考虑了与对手机器人相顶时的负载电流,TA7257P 也是足够用的。想控制更大的电机电流时,还有同样为东芝公司制的 TA8429H 及 SGS Tomson 制的 L6203 等。电机驱动 IC 上还装有散热片。

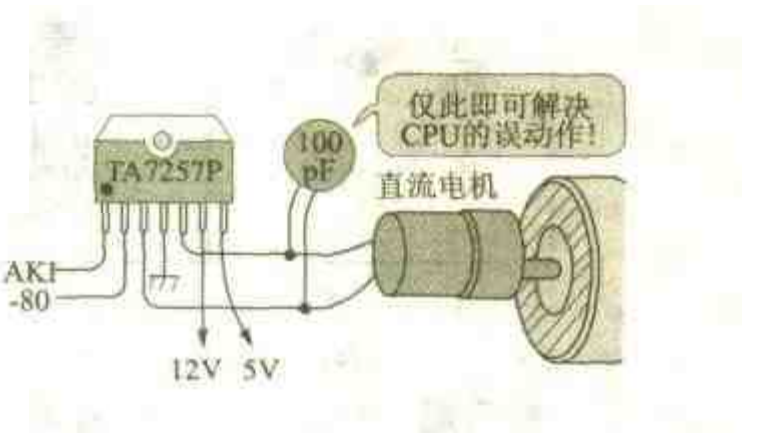

TA8429H, L6230

如果使用专用 IC,直流电机的控制能变得简单,是不是?

可是,在使用直流电机时,微机板常有误动作。其原因是,直流电机旋转时电刷产生的电噪声会给微机板以不良影响。

怎么能解决这一问题?

给电机的输入端接上 100pF 的小电容试一试。用这种方法大体上可以解决。



仅仅一只电容就能防止误动作。

其他电路

除了上述的主要电子电路部分外,以下的电路也是必需的。

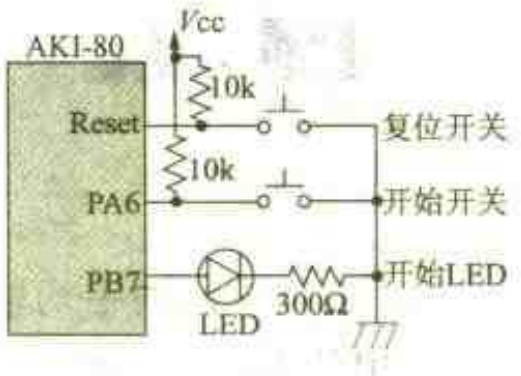
其他电路

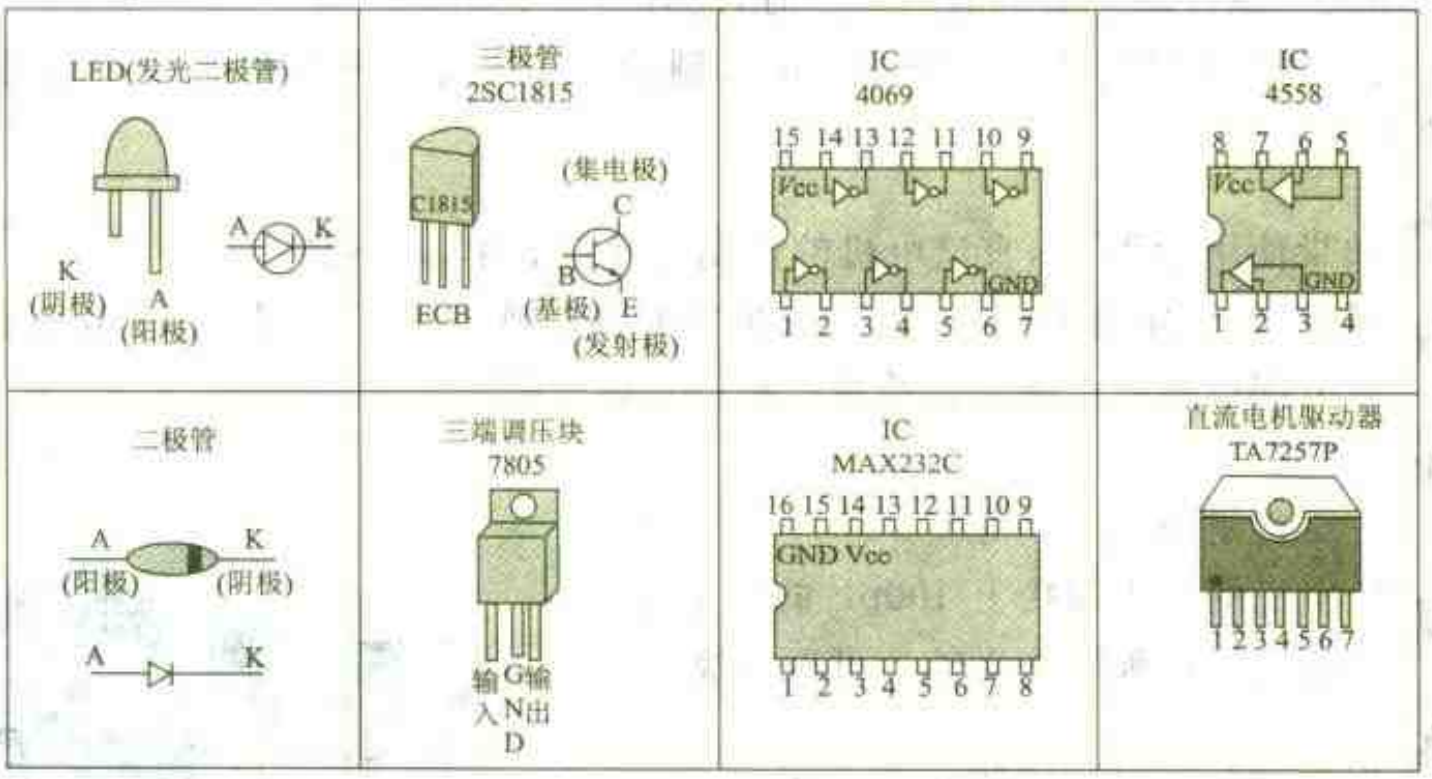
① 复位开关
　使微机板复位时使用。
② 开始开关
　使“太郎”开始工作时使用。
③ 开始 LED
　按下开始开关后直至太郎开始动作前的 5 秒钟时间监视时使用。

全部电路并不是一齐组装，而是分部分组装，确认工作正常后再组装下一部分。

以上是全部电子电路的介绍。请用 P140 的电路图对原理进行确认。

下图示出了所使用的半导体器件的引脚配置。





练习题

●组装“太郎”的电子电路时都有哪些注意事项。

解答

为防止静电，应使用陶瓷绝缘的小型电烙铁；
充分考虑各部分的配置，要平衡良好；
确认各部工作正常的同时再进一步制作；
不要弄错 IC 等的引脚配置等。



6 执行文件的制作方法



程序语言


微机板用的执行程序是用个人计算机作成的。


但是执行程序最终要写入 ROM,装在微机板上。


是的。下面介绍为写入 ROM 制作执行程序的顺序。使用 ROM 写入器将执行程序写入 ROM 的顺序在下节介绍。




使用的程序语言是什么?


介绍以下 3 种语言的使用例子。

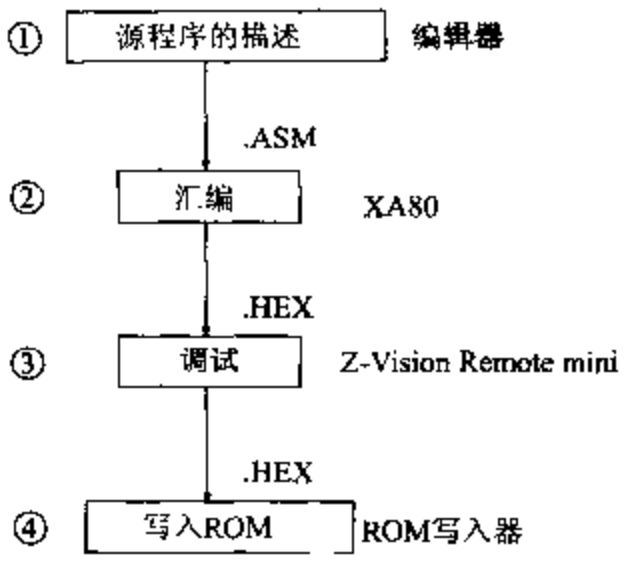
程序语言

① 机器语言
② C 语言
③ BASIC

具体使用的是 System Load 有限公司发售的下述版本。

① XA80 Ver. 2. 1
② MINI C Compiler Ver. 1. 0(限定版)
③ Z80BASIC 编译器

以上无论哪一种都能以合适的价格从秋月电子通商有限公司买到。使用环境设定为 PC98 系列 MS-DOSVer. 3. 1 以上。



机器语言



用机器语言编写执行程序时的流程如右图所示。

用机器语言编制执行程序的顺序

① 使用编辑器记述程序。作为扩展名为. ASM 的文本保存；
② 用 XA80 汇编程序汇编。作成扩展名为. HEX 的执行文件；
③ 用 Z-VISION mini 调试器调试；
④ 用 ROM 写入器将程序 ROM 化。
(③和④在后面的章节中介绍)

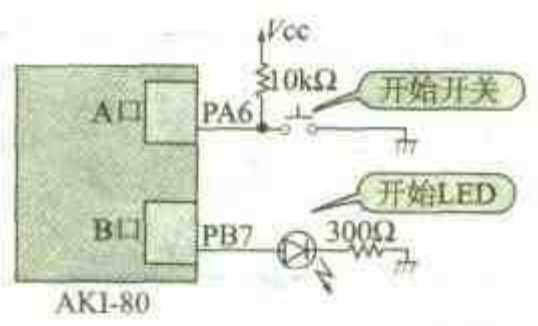


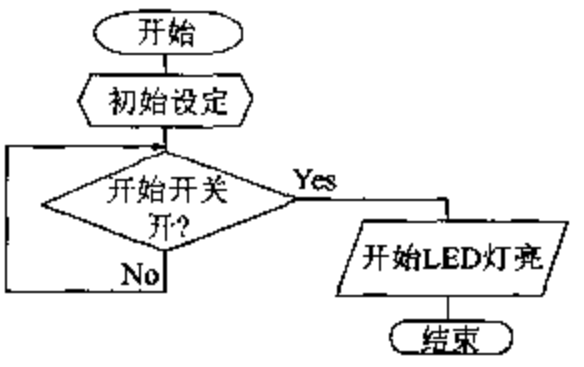
请列举具体的例子介绍吧。



好吧。前面制作的“太郎”的电路中，PIO、A 口的 PA6 与开始开关连接，B 口的 PB7 与开始 LED 连接。

所以，就以“按下开始开关，开始 LED 灯亮”的程序为例，来看看执行程序的编写。



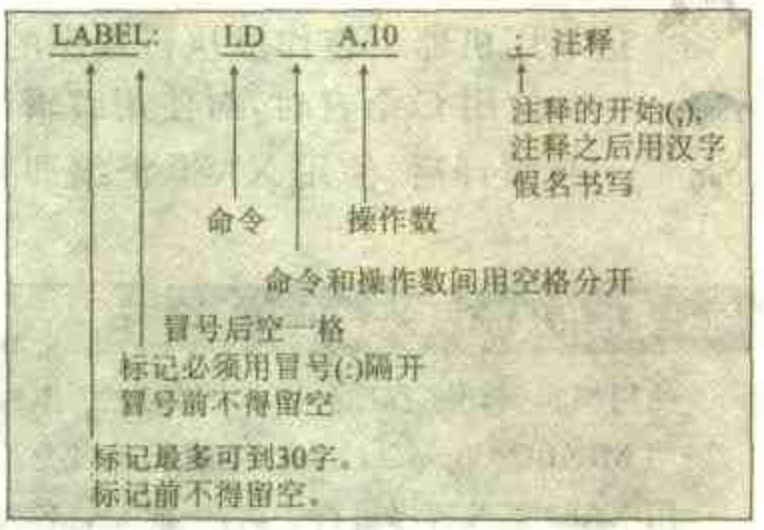


机器语言的源程序如下：

```
        ORG     0000H ——开始地址
        LD      A,4FH
        OUT     (1DH),A      A口输入设定
        LD      A,0FH
        OUT     (1FH),A      B口输出设定
        LD      A,00H
        OUT     (1EH),A      B口清除
ST_SW:  IN      A,(1CH)
        AND     40H          PA6输入
        JP      NZ,ST_SW     -PA6判断
        LD      A,80H
        OUT     (1EH),A      PB7输出
        END
```

那么来看看具体的步骤。

① 用编辑器记述软件程序。编辑器除 MIFES 等之外，也可使用文字处理机软件。保存文件时，指定文本形式取扩展名为. ASM。而且，XA80 的程序书写格式定为以下形式：



详细情况，请参照 XA80 用户说明书。

② 使用 XA80 对以上作成的文件进行汇编。如无书写格式错误，即可生成扩展名为. HEX 的执行程序。

```
A>XA80
Z-80/64180 Absolute Cross Assembler (MS-DOS) Verision 2.01
Copyright(C)System Load CO.,LTD. 1989－1991 All rights
reserved.
    Source  File[.ASM]:LED1
    Object  File[.HEX]:
    Symbol  File[.SYM]:
    Listing File[.LST]:
    Pass 1...
    Pass 2...
                XA80 起动屏幕
    No Errors
    A>
```

作成的 HEX 文件即是可写入 ROM 的执行程序。

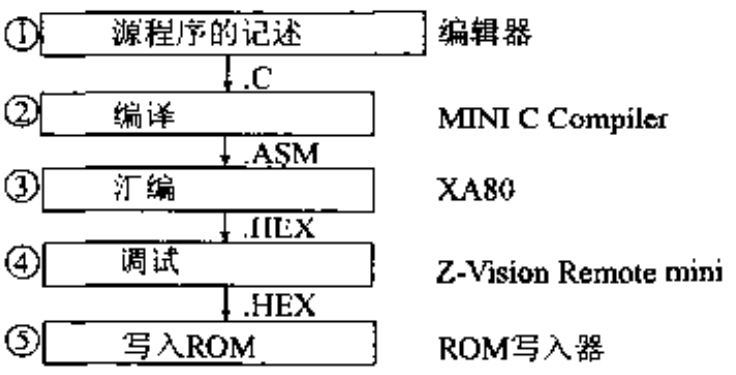
这里所介绍的是直到作成 HEX 文件为止。之后所必需的调试、ROM 化

顺序等在后面的章节介绍。



到那时，这里所作的执行程序的工作即可确认，是吧。

C 语言





C 语言是作为能像机器语言一样进行极细致处理的高级语言而广泛普及的程序语言。使用 MINI C Compiler，即使不具备机器语言的知识，也能用 C 语言开发微机板用的程序。用 C 语言作成执行程序时的工作流程如右图。



这与用机器语言作成执行程序的工作顺序相似。



使用 C 语言时，对使用编辑器记述的 C 语言源程序用 MINI C Compiler 编译后，使用 XA80 汇编即可。

用 C 语言作成执行程序的工作顺序

① 使用编辑器记述程序。以扩展名为.C 作为文本文件保存；
② 用 MINI C Compiler 编译，生成扩展名为. ASM 的文件；
③ 用 XA80 汇编程序进行汇编，生成扩展名为. HEX 的执行文件；
④ 用 Z-VISION mini 调试器进行调试；
⑤ 用 ROM 写入器 ROM 化。
（④、⑤在后面的章节中介绍。）

与用机器语言同样，使用“按下开始开关，开始 LED 灯亮”的程序例子，试编成执行程序。



C 语言的源程序如右图所示。

```
#include "k15c.h"          开始地址用头文件
main ()                    STARTUP.H指定
{
    char ad;
    out(0x1d,0x4f);        ——— A口输入
    out(0x1f,0x0f);        ——— B口输出

    out(0x1e,0x00);        ——— B口清除
    ad=1;
    while(ad!=0){
            ad=in(0x1c);
            ad=ad & 0x40;  ——— PA6输入
    }
    out(0x1e,0x80);        ——— PB7输出
}
```



下面介绍具体的步骤：

① 用编辑器记述上边的源程序。以扩展名为.C保存；

② 使用MINI C Compiler对上面作成的文件进行编译。如无错误，即可作成扩展名为.ASM的文件。

```
A>XA80 testc.c
System Load Compiler [mini-'C' V1.0 for Z80 KC80 1995.3
By System Load CO.,LTD. & K.Kino Copyright (C) All rights reserved.
There were 0 erros in compilation.
A>
```

MINIC编译器起动屏幕

③ 使用XA80对上述作成的文件进行汇编。书写格式等无错误时，即可作成扩展名为.HEX的执行程序。

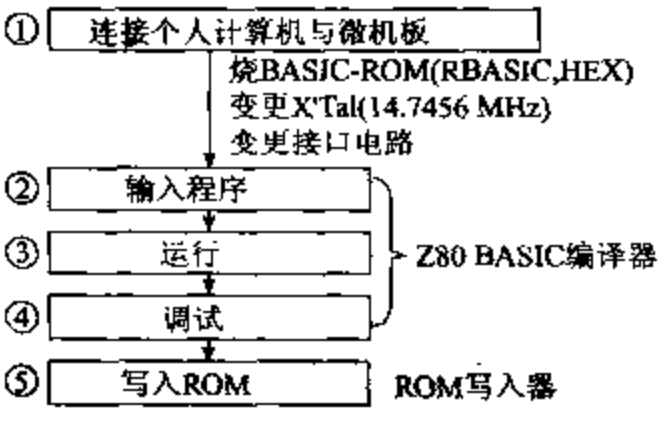
```
A>XA80 testc.asm
Z-80 64180 Absolute Cross Assembler (MS-DOS) Version 2.01
Copyright(C)System Load CO.,LTD. 1989—1991 All rights
Object File[.HEX]:
Symbol File[.SYM]:
Listing File[.LST]:
Pass 1...
Pass 2...
No Errors
A>
```

XA80起动屏幕

BASIC



BASIC是作为初学者使用的对话型高级语言而更一般广泛普及的程序语言。使用Z80BASIC编译器，即使不具备机器语言的知识，也能用BASIC开发微



机板用的程序。

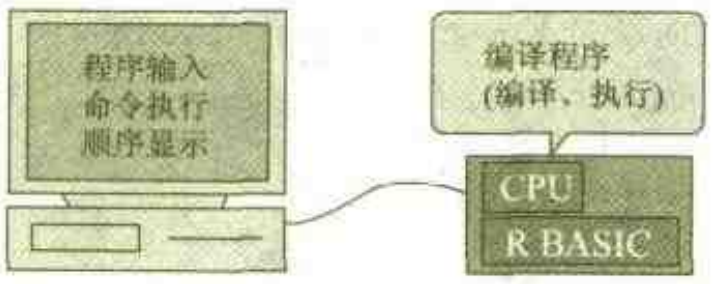


为了使用 BASIC，要在微机板上安装 BASIC-ROM，用 RS-232C 连接个人计算机与微机板。



是的。要在与微机板通信的同时作成 BASIC 程序。并将 Z80BASIC 编译程序中所包含的 RBASIC. HEX 写入 BASIC-ROM。

写入方法请参照下节。处于通信状态的个人计算机与微机板分担的工作任务如下。





至今记述程序使用的是外部编译器，而 BASIC 中就包含有编译器。

用 BASIC 作成执行程序的工作顺序

① 连接个人计算机与微机板；
② 使用 BASIC 编译器记述程序；
③ 执行程序确认动作；
④ 如有错误进行调试排除；
⑤ 用 ROM 写入器 ROM 化。



如果使用 BASIC 编译器，就不必使用 Z-VISION mini 调试器。



使用至今相同的“按下开始开关、开始 LED 灯亮”的程序例子，试作执行程序。

BASIC 的源程序如下：

```
100 out $ 1d $ 4f          ——— A口输入
110 out $ 1f $ 0f          ——— B口输出
120 out $ 1e $ 00          ——— B口清除
130 a=inp ($1c)
140 a=a and $ 40           ——— PA6输入
150 if a<>0 goto 130       ——— PA6判断
160 out$1e $ 80            ——— PB7输出
170 end
```

① 连接个人计算机与微机板；

② 输入上述源程序：

```
                    BTERM
                                        {System Load}
■■■■■■■■■■ BASIC Terminal program ■■■■■■■■■■■■
>
>list
   100 out $ 1d $ 4f
   110 out $ 1f $ 0f
   120 out $ 1e $ 00
   130 a=inp ($1c)
   140 a=a and $ 40
   150 if a<>0 goto 130
   160 out$1e $ 80
   170 end
ok
>
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
BTERM for PC-9801
Copyright(C)Systems Load CO.,LTD.1992.All
right reserved.
Initialized:Baud rate is 9600
      Quit  Load  Lload    Run  Cont  List Wrt Blank
```

Z80BASIC
编译程序起动屏幕

③ 执行程序：

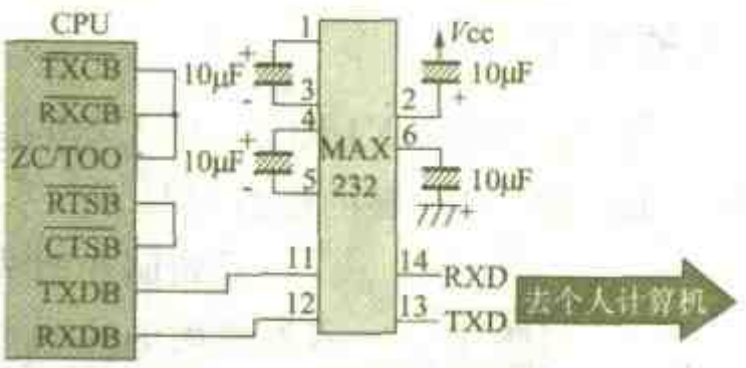
命令
程序的执行 f-6 键
执行中程序的中断 SHIFT + f-6 键

④ 根据需要进行调试：

命令
程序的ASC形式保存
SHIFT + f-3 键

⑤ 连接 ROM 写入器，将程序写入 ROM。

命令
将存储器中的程序与BASIC系统一起写入ROM　SHIFT + f-9 键



连接个人计算机的接口例

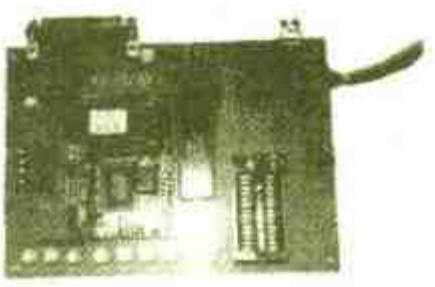
7 ROM写入器的使用方法



ROM 写入器

ROM 写入器是将用个人计算机编成的程序写入 ROM 的装置。

现在市面上出售的 ROM 写入器种类非常多，功能与价格也各种各样。这里，使用秋月电子通商有限公司销售的 ROM 写入器套件来介绍其使用方法。


ROM写入器

套件要自己组装吗?

是的。但是组装很简单不必担心。

AKI-ROM 写入器的制作

此 ROM 写入器套件(以下称 AKI-ROM 写入器)是能以合适的价格购得，且对于我们的用途功能也足够的推荐制品。



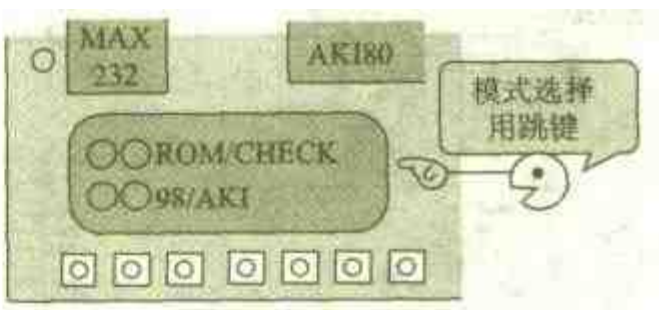
开始制作时先阅读 AKI-ROM 写入器所附的说明书，然后细心制作。

AKI-ROM 写入器中带有 AKI-80 微机板。

AKI-ROM 写入器由 AKI-80 控制。所以使用 AKI-ROM 写入器时必须要有一个 AKI-80 微机板。AKI-ROM 写入器套件中，有包含 AKI80 的，也有要另买的，所以要注意。

AKI-ROM 写入器中备有校核模式。制作结束时要立即校核。校核完后，将模式选择跳键置为“开”。

模　式	ROM/CHECK	98/AKI
98 控制 ROM 写入器	开	开
AKI 控制 ROM 写入器	开	短接
校核/文件传送	短接	短接



<AKI-ROM写入器>

这里采用 PC98 系列 MS-DOS 并利用 AKI-ROM 写入器。

个人计算机用软件的准备

为了让 AKI-ROM 写入器工作，个人计算机和 ROM 写入器两者都要有通信用的软件。这些软件写在套件所含的 AKI-80ROM 中。



个人计算机用的通信软件，需由此 ROM 中拷贝到软盘上，即可使用。

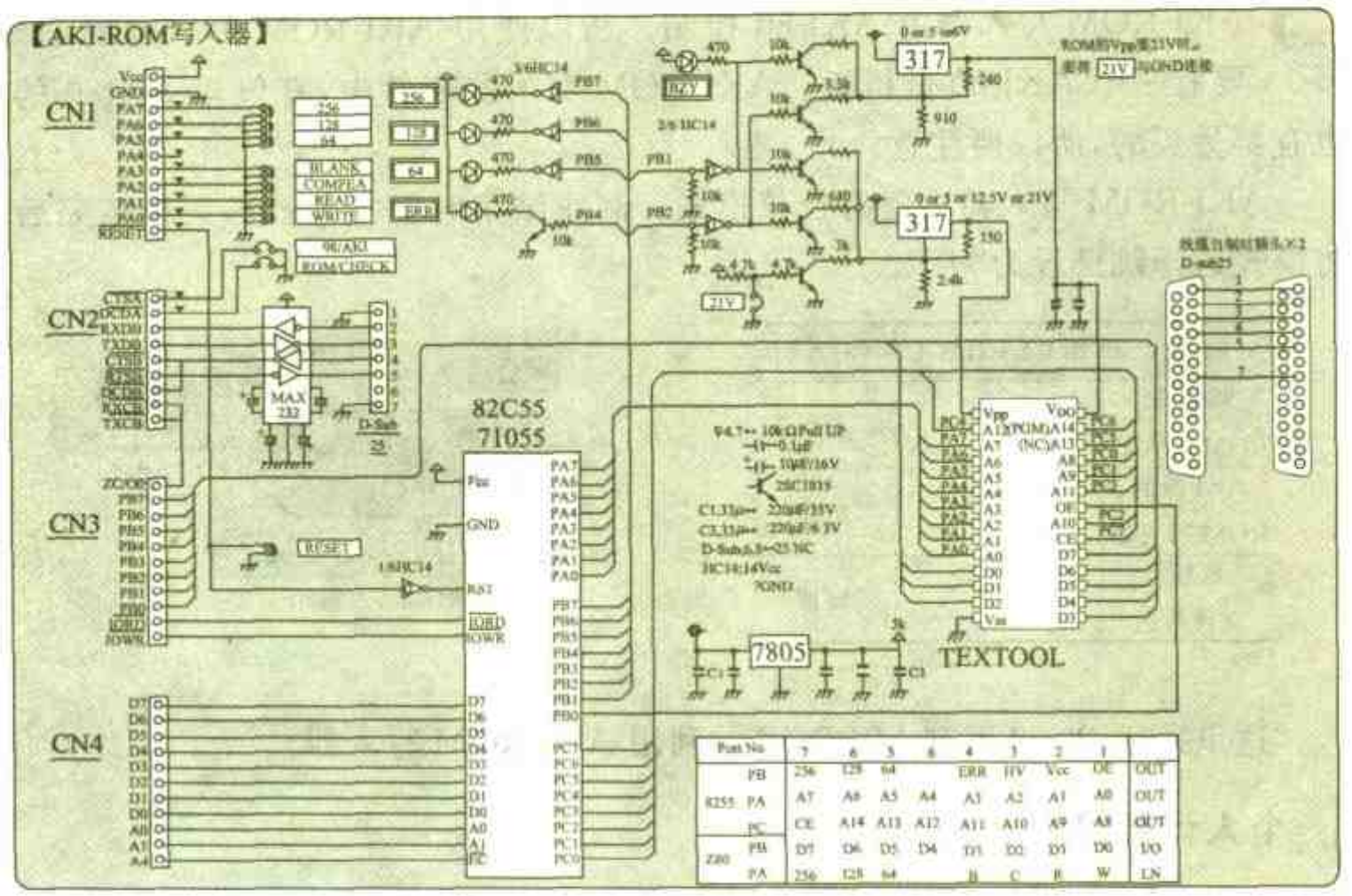
是的。个人计算机与 AKI-80ROM 写入器通过 RS-232C 光缆连接进行工作。工作顺序详细说明于 AKI-ROM 写入器的操作说明书中，请遵照执行。个人计算机用的通信软件的名字为 AKI ROM. EXE。

ROM 写入器的使用方法

现在使用 AKI-ROM 写入器给 ROM 中写入程序。ROM 使用日立制的 HN27

C256AG-12。此 ROM 写入时需要 12.5V。这个电压要 AKIROM 写入器供给，所以要将 15～18V 的电源接给 AKI-ROM 写入器。







先来确认用 AKI-ROM 写入器将程序写入 ROM 的工作步骤。

将程序写入 ROM 的步骤

① 用 RS-232C 光缆连接个人计算机与 ROM 写入器；
② 打开个人计算机和 ROM 写入器的电源；
③ 在个人计算机上启动 AKIROM. EXE 程序；
④ 指定 ROM 的类型；
⑤ 将想要写入的程序经软盘等方式读入个人计算机的内存；
⑥ 将程序传送给 ROM。



AKI-ROM写入器

那么，实际来操作一下看看。将前一节中用机器语言作成的“按下太郎的开始开关，开始 LED 灯亮”程序写入 ROM




```
ORG       0000H
LD        A,4FH
OUT       (1DH),A
LD        A,0FH
OUT       (1FH),A
LD        A,00H
OUT       (1EH),A
ST_SW:IN  A,(1CH)
AND       40H
JP        NZ,ST_SW
LD        A,80H
OUT       (1EH),A
END
```

TEST.ASM



仿照下面给出的“ROM 写入实例”进行作业。

将写入了程序的 ROM 装到“太郎”的 AKI-80 上，即可让程序运行。

ROM写入实例

① 确认个人计算机与ROM写入器的连接,打开电源,在个人计算机上启动通信软件AKIROM:

AKI-80 ROM WRITER ver 1.2		秋月电子通商
ROM TYPE	OPERATE COMMAND SELECT	DATE DIR=A:¥
2764 27128 27256 2864	Load HEX File Write & Transmit Write from AKI-RAM Transmit to AKI-RAM Read & Receive Read to AKI-RAM Compare HEX File Compare AKI-RAM Blank Check Rom Type Start Stop Address Save HEX File	HEX FILE=*.HEX AKIROM.EXE 起动屏幕
STRT ADS STOP ADS		
0000 H 7FFF H		

[ESC] to EXIT

② 将未使用的ROM装到ROM写入器的TEXTOOL(带杠杆的ROM插座)上:



TEXTOOL

③ 在个人计算机上选择型号为27256 的ROM:



④ 选择OPERATE COMMAND SELECT中的Load HEX File,将TEST.HEX读入个人计算机的内存:



⑤ 选择OPERATE COMMAND SELECT中的Write&Transmit,将个人计算机内存中的程序TEST.HEX传送给ROM。

ROM 的拷贝



能对写有程序的 ROM 进行拷贝吗？



可以。ROM 的拷贝按以下的步骤进行。

ROM 拷贝的步骤

① 按照与前相同的步骤，启动 AKIROM. EXE；

② 将拷贝源 ROM 装在 ROM 写入器的带杠杆的插座上；

③ 选择 OPERATE COMMAND SELECT 中的 Read & Receive，将 ROM 中的程序读入个人计算机的内存中；

④ 将要烧的 ROM 换装在 ROM 写入器的带杠杆的插座上；

⑤ 选择 OPERATE COMMAND SELECT 中的 Write & Transmit，将个人计算机内存中的程序传送给 ROM。


拷贝重要的 ROM 时，只要预先做好准备即可防止发生故障。

其他指令


AKIROM. EXE 中，还准备有检查程序是否正确写入 ROM 等指令。

Compare HEX Files

将读入个人计算机内存中的 HEX 文件与 ROM 的数据进行比较。

Compare AKI-RAM

将 ROM 写入器上的 RAM 数据与 ROM 数据进行比较。

Blank Check

检查 ROM 是否空白(可写入状态)。

ROM 擦除器

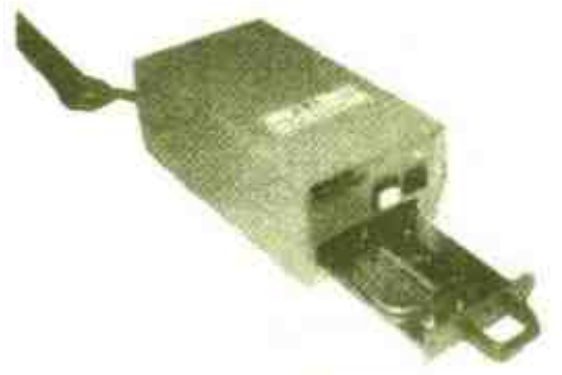

写了程序的 ROM，要贴上封条，以免光(紫外线)进入窗口。

为什么 ROM-IC 上带有窗口呢？这是因为将窗口对着紫外线即可消去 ROM 中的数据。


就是说，如果将 ROM 对着紫外线消去数据，就能写入不同的数据再重新使用。


是的。将 ROM 对着紫外线以消去其数据的装置，即为 ROM 擦除器。市面上有各种型式的 ROM 擦除器销售。买一个带定时器的便宜的擦除器，使用非常便利。


自然的太阳光中也含有紫外线。所以，为防止写好了的 ROM 数据消失，而要用封条盖上窗口。


ROM擦除器

练习题

●对于写好了的 ROM，能否重新写入新的数据？

解答

不能。应该用 ROM 擦除器擦去后再写入新的数据。

8 遥控调试程序的使用方法



程序的调试



前面介绍了编写相扑机器人用程序的3种程序语言(机器语言、C语言、BASIC语言)。当然,使用哪种语言是因人而异的。下面介绍使用机器语言编制程序的顺序。

在第3章的"微机控制的顺序"节中,已介绍过两种程序ROM化的顺序。



是的。"顺序1"是修正程序时写入ROM并进行校核的方法。"顺序2"则是将修正过的程序传给RAM并进行动作校核后再写入ROM的方法。

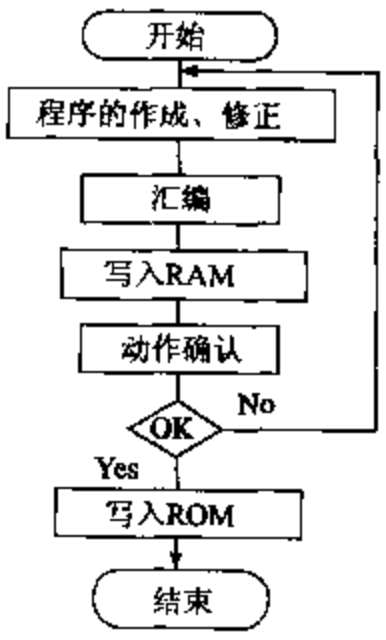


比起多次使用ROM写入器及擦除器的"顺序1",最终确认动作后只烧1次ROM的"顺序2"有很好的效率。



所以下面介绍用"顺序2"的方法进行程序调试的方法。先来确认"顺序2"的工作流程。

如能理解下面的流程图,学习起来就容易很多。



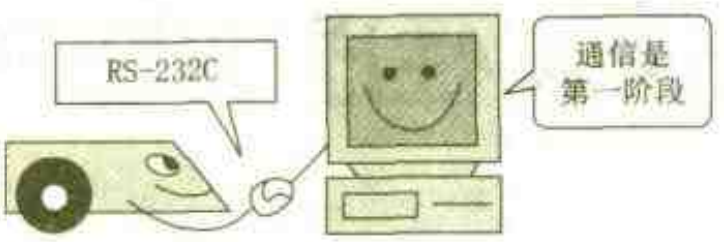
程序ROM化的顺序2

遥控调试程序



请具体教我们实际的方法。

好的。最初的目的是要实现相扑机器人携带的 AKI-80 微机板与个人计算机间的通信。



通信的准备

① 用 RS-232C(线缆)连接相扑机器人与个人计算机；
② 将写有通信用程序的 ROM 装到相扑机器人(AKI-80)上；
③ 在个人计算机上启动遥控调试程序。

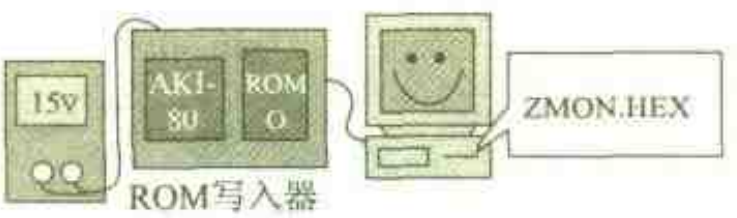
本书使用的遥控调试程序为 Z-Vision Remote mini(System Load 公司制)。此调试程序是 Z-Vision Remote 产品的功能限定版。与原版相比，虽然有可使用的指令数目少等限制，但对于初学者的使用功能已足够。它的价格只有数千日元，已经是很好的产品了。

怎样准备②中要用的通信 ROM?

Z-Vision Remote mini 的软盘中含有以下文件(文件名因版本而异，请参照使用说明书)。

Z-Vision Remote mini Ver. 3. 0 的文件

ZVRMINI. EXE
　　个人计算机使用的 Z-Vision Remote mini 程序。
ZMON. HEX
　　TMP-Z84C015 用的写入 ROM 的通信程序。
SPMON. HEX
　　SUPER AKI-80 用的通信用程序。



这里，将 ZMON. HEX 写入 ROM 来使用。

要将程序写入 ROM，使用前面介绍过的 ROM 写入器即可。

是的。请参照第 4 章“ROM 写入器的使用方法”制作通信用 ROM。而且，个人计算机启动的遥控调试程序为 ZVRMINI. EXE。为使用 RS-232C 接口，要在个人计算机中设定名为“RSDRV. SYS”的驱动文件(含于 MS-

DOS的系统盘中)。不要忘记将此驱动文件置于根目录下,如下所示记在CONFIG.SYS中。

CONFIG.SYS上增加1行

```
DEVICE=RSDRV.SYS
```

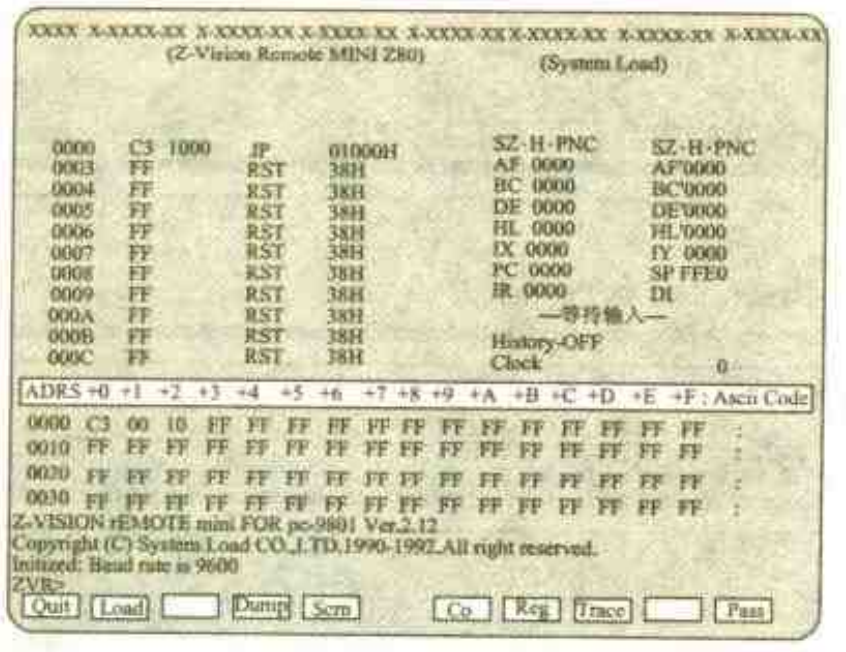
(注)ZVRMINI.EXE只能在PC-9801的MS-DOS上运行。要在其他的机种上使用时,要准备Z-Vision Remote的原版。



至此,相扑机器人(AKI-80)与个人计算机的通信的准备工作完毕。



那就立刻试试看。如上所述,连接相扑机器人与个人计算机,打开相扑机器人的电源,从个人计算机MS-DOS的提示符A(A>)开始,打入ZVRMINI启动Z-Vision Remote mini。这样一做,即会显示以下的画面。





如果在画面的下方显示“ZVR>”呈等待指令输人状态,即可准确进行通信。如果无画面显示,或出现“不能通信”等错误信息时,应检查电缆的连接、与AKI-80通信程序的时钟一致,以及MS-DOS上RS-232C的设定等。

Z-Vision Remote mini的画面



Z-Vision Remote mini画面上显示有很多信息哩。



这些信息都与AKI-80的存储器及CPU有关。

Z-Vision Remote的原版准备有大约50个指令,但在mini版中能够使用的只有其中的11个。可以说许多功能受到了限制,但对于调试相扑机器人的程

序则是功能很充分的，因此无需担心。

下面对其指令做一简单说明。

Z-Vision Remote mini 的指令一览表

指　令	功　能
A	进行汇编
BP	断点设定
BC	断点删除
D	存储器的显示与变更
G	程序运行
L	HEX 文件及符号文件的装入
P	路径跟踪
Q	结束 Z Vision Remote mini
R	寄存器值的设定
T	跟踪
U	反汇编列表显示

使用特别频繁的，大概是以下 5 个指令：

D 存储器的显示与变更

将从指定地址开始的存储器的内容显示在打印窗口。

G 程序运行指令

从指定的地址开始，运行 AKI-80 程序。

L 程序・文件的装入

将 HEX 形式的文件从软盘读入内存。

Q 结束

结束 Z-Vision Remote mini。

U 显示反汇编

将反汇编程序(助记符)显示在显示窗口中。

几条指令也可以由功能键来分担。

光标向显示窗口、打印窗口等的移动，使用功能键 4“Dump”和 5“Scrn”来进行。



试着使用一下显示存储器内容的 D 指令。例如，要显示从地址 0050 开始的存储器的内容，那就要输入成

```
ZVR>D0050
```

Z-Vision Remote mini 画面的说明

(画面可分为 7 个区,这里只使用其中一部分。)

① 对话窗口

指令输入画面。跟在 ZVR≫后打入指令。也有显示来自 Z-Vision Remote mini 的信息的。

② 显示窗口

用十六进制数与助记符两者显示 AKI-80 存储器(ROM、RAM)中的内容。

③ 寄存器窗口

显示 AKI－80CPU 的寄存器、标识位的状态。

④ 信息打印窗口

用十六进制数显示 AKI-80 存储器(ROM、RAM)的内容。

这里可对存储器的内容进行编辑。当然可编辑的仅限于 RAM 的内容。

⑤ 路径断点窗口

显示断点(AKI-80 程序运行途中的停止点)的设定状态。

⑥ 信息窗口

显示以下三种信息:

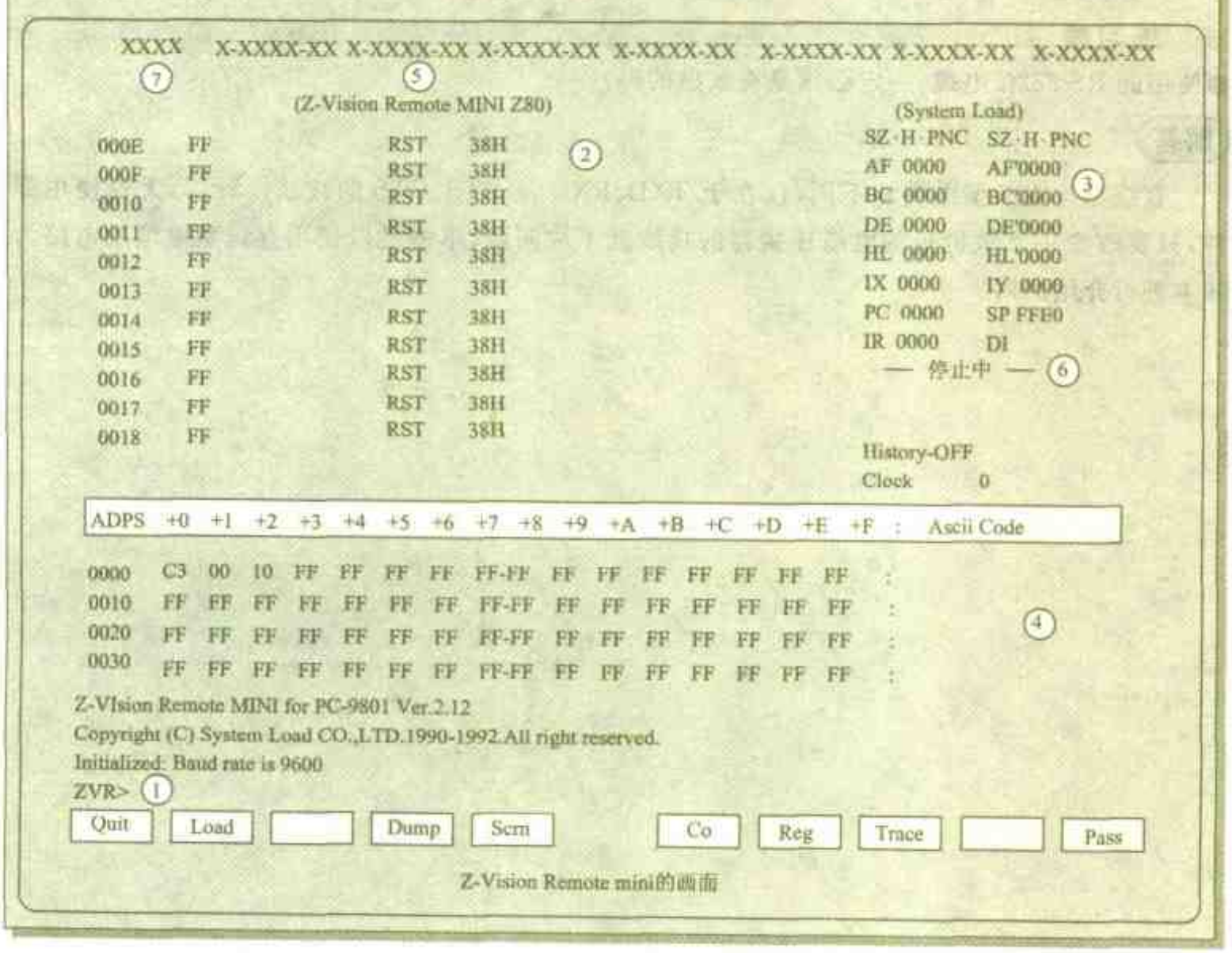
通信中——正在发送接收数据,

停止中——等待指令输入,

实行中——AKI-80 程序正在运行。

⑦ 断点窗口

显示即时断点。



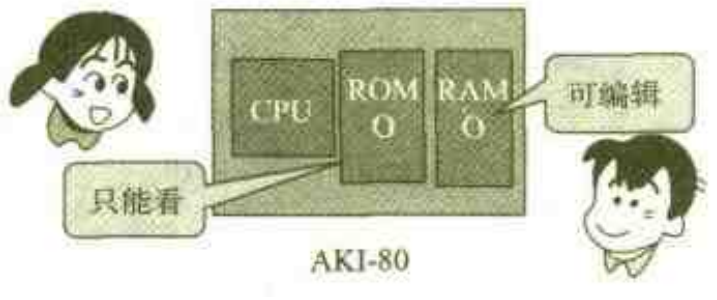
Z-Vision Remote mini的画面



打印窗口处即显示从地址 0050 开始的存储器的内容。用功能键 4 将光标切换到打印窗口，上下移动也可以改变显示地址。



现在，两个人要显示的是写入 AKI-80 所带 ROM 中的通信程序的内容。此程序写在 ROM 中，所以尽管看得见，但要修改及删除等编辑都不行。Z-Vision Remote mini 能够编辑的仅限于写在 AKI-80RAM 中的程序。





这样一来，使用遥控调试程序对相扑机器人的程序进行调试的准备就做好了。

总之，可以一边看着个人计算机的屏幕，一边编辑 AKI-80 的存储器的内容（相扑机器人的程序）。

练习题

●使用的 RS-232C 电缆，一定必须是直线型的吗？

解答

直线型与交叉型电缆的不同，仅在于 TXD、RXD 及 RTS、CTS 的接法。所以，无论使用哪种，只要改变一下微机板的电缆连接器的接线就不成问题（本书是以使用直线型电缆的电路为基本进行介绍的）。

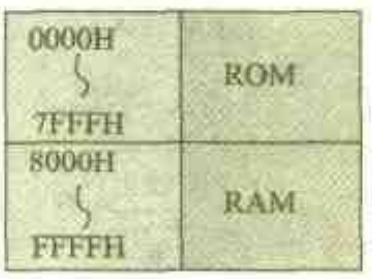
9 程序调试实践



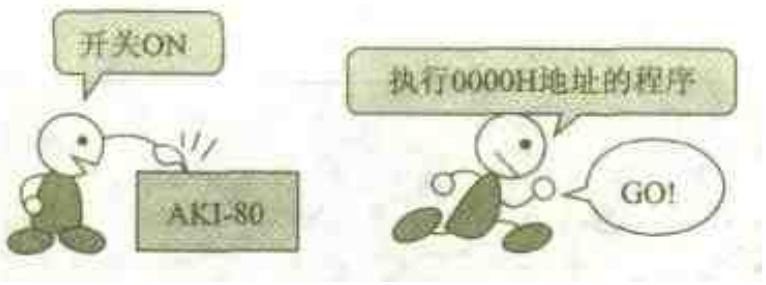
关于存储器的地址



AKI-80 微机板上可安装 ROM 和 RAM 两者。两个存储器 IC 的地址如下(参照第 4 章第 3 节“微机板的使用方法”)。







这里有一点必须注意,就是在接通微机板的电源时,或其后按了复位开关时,CPU 会从存储器的最前面的地址(0000H)开始执行程序。



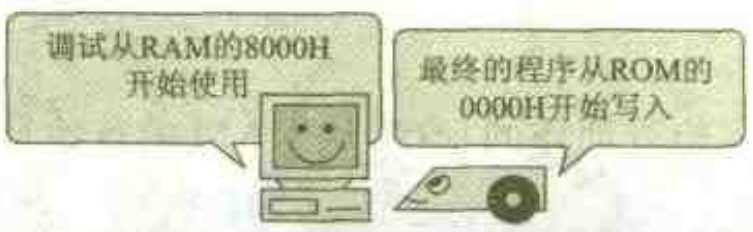
就是说,最终作成的相扑机器人的程序从 ROM 的 0000H 地址开始写入。



是的。但是用遥控调试程序调试好的仅仅是写入 RAM 的程序。所以,相扑机器人的程序的最前面的地址是 8000H。



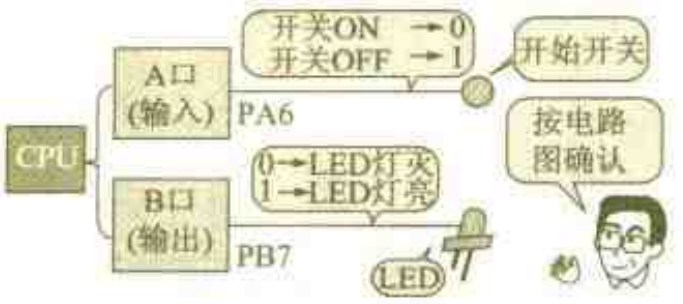
用遥控调试程序对程序进行检查时,是从 8000H(RAM 的首地址)开始写入程序,而烧 ROM 时变为 0000H(ROM 的首地址)即可。



LED 灯亮程序



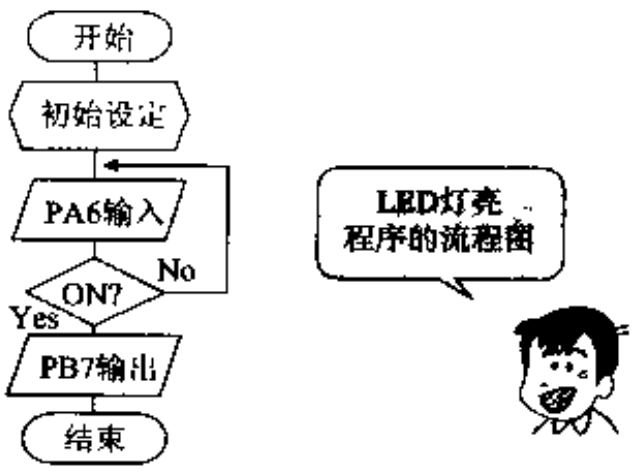
实际来操作一下程序对相扑机器人控制。“太郎”的开始开关接在 PA6 上，LED 接在 PB7 上。



利用此电路，试编写“按下开始开关，LED 灯亮”的程序。



考虑程序的流程图为





以流程图为基础编写程序。

```
        ORG     8000H       ;地址设定
        LD      A,4FH       ;A 口输入
        OUT     (1DHA),
        LD      A,0FH
        OUT     (1FH),A     ;B 口输出
        LD      A,00H
        OUT     (1EH),A     ;B 口清除
INKEY:  IN      A,(1CH)     ;A 口输入
        AND     40H         ;PA6 屏蔽
        JP      NZ,INKEY
        LD      A,80H       ;PB7 输出
        OUT     (1EH),A
        END
```

LED 灯亮程序



用遥控调试程序检查动作的程序，从 RAM 的首地址(8000H)开始书写。



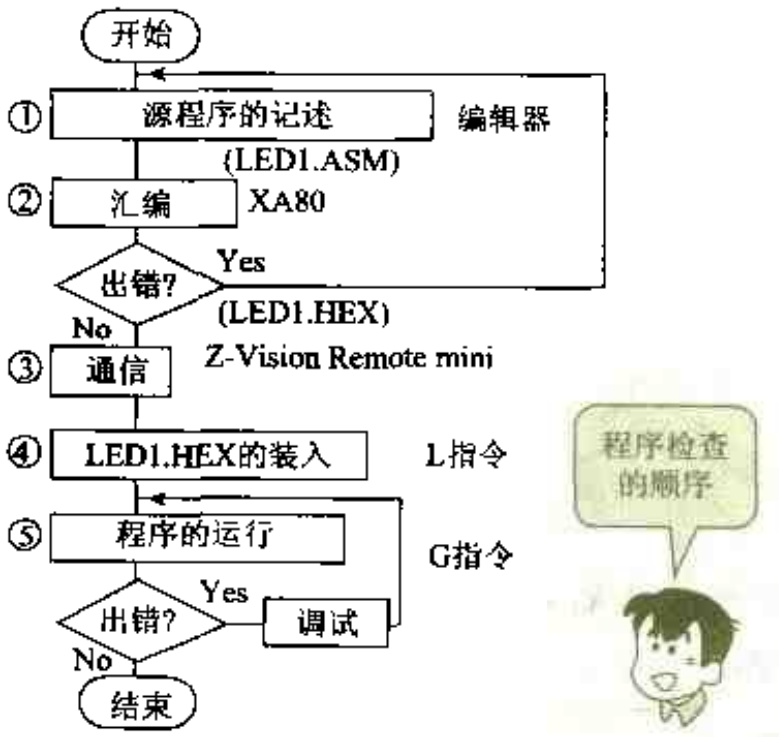
现在来说明直至让程序运行的顺序。

程序检查的顺序

① 使用编辑器记述程序，以文本形式的文件保存；
② 将上面作成的文件用 XA80 汇编，作成 HEX 形式的文件；
③ 使个人计算机和相扑机器人成为可通信状态；
④ 用遥控调试程序读入②作成的文件。读给遥控调试程序的同时也传送给相扑机器人的 RAM；
⑤ 运行程序。



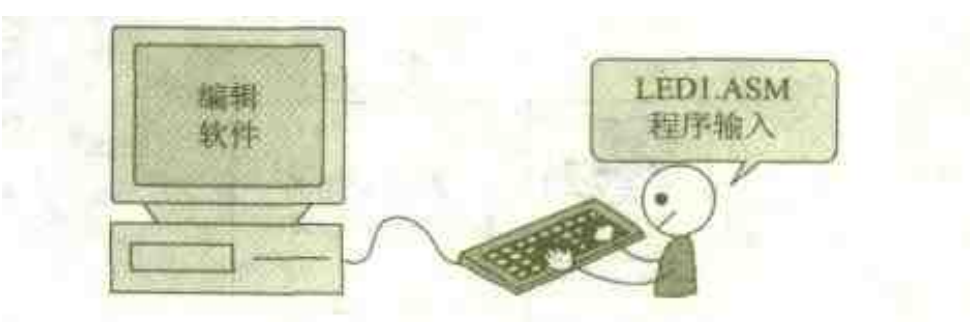
如前面所讲的，如果使用遥控调试程序，无需 ROM 的介入即可检查程序的动作，因而很便利。





立刻按此顺序进行实际工作试试。

① 使用编辑软件输入程序，用 LED1. ASM 名字保存；



② 用 XA80 对 LED1. ASM 进行汇编，生成 LED1. HEX；

```
A>XA80
Source      File[. ASM]:LED1
Object      File[. HEX]:
Symbol      File[. SYM]:
Listing     File[. LST]:
Pass 1 …
Pass 2…
No Errors
```

③ 连接个人计算机与相扑机器人，启动 Z-Vision Remote mini；

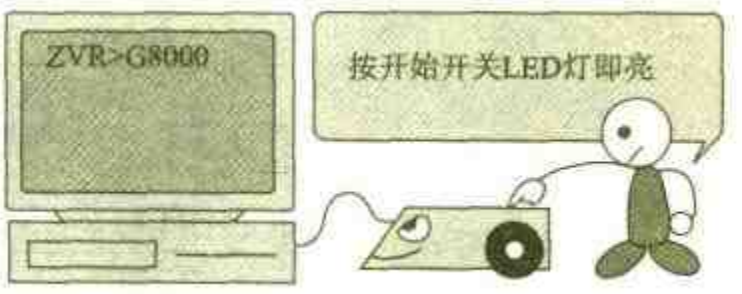
④ 用 Z-Vision Remote mini 读入 LED1. HEX；




```
ZVR>L
File name[. HEX]   =LED1
Load HEX file 'LED1. HEX' 8000-8015
File name [. SYM]  =
```

⑤ 用 Z-Vision Remote mini 让程序运行。此时要指定程序运行的地址从 8000H 开始。

只要一按相扑机器人的开始开关，LED 灯即亮。

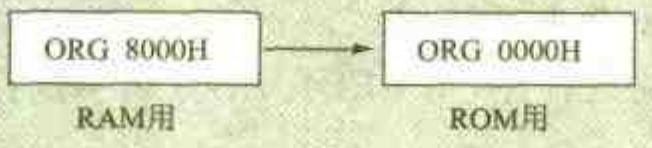




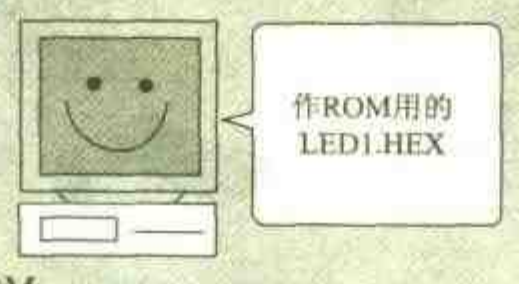
做得很好。程序完成后，将程序的开头地址改为 0000H 写入 ROM 即可。

程序写入 ROM 的步骤

① 使用编辑程序修改程序(首地址)；



② 用 XA80 汇编；



③ 使用 ROM 写入器写入 ROM。



LED灯亮程序顺利运行了。但是，如果程序中有故障时该怎样调试排除呢？



程序中有故障时，有两种修正方法可以考虑。如果程序修改的量很大时，可使用编辑程序改写程序。即从最初开始再重复一遍程序检查的步骤。另外，如果程序稍作修改即可的话，就在操作 Z-Vision Remote mini 的同时进行调试。这里实验的程序中没有故障，但可以通过更改此程序来练习调试的步骤。

LED灯灭程序



刚才的程序动作是“按下开始开关，LED灯亮”。现在将此改为“按下开始开关，LED灯灭”程序试试。

```
        ORG     8000H       ;地址设定
        LD      A,4FH       ;A口输入
        OUT     (1DH),A
变更    LD      A,0FH       ;B口输出
        OUT     (1FH),A
        LD      A,80H       ;LED灯亮
        OUT     (1EH),A
INKEY:  IN      A,(1CH)     ;A口输入
        AND     40H
        JP      NZ,INKEY    ;PA6屏蔽
        LD      A,00H       ;LED灯灭
        OUT     (1EH),A
        END
```

LED灯灭程序



程序的变更有两处。

变更①：最初置LED灯亮。

变更②：按开始开关时，LED灯灭。



再来说明一下修改程序的步骤。

程序修改(调试排除故障)的步骤

① 用 Z-Vision Remote mini 读入刚才的 LED1. HEX 文件；

```
ZVR>L
File name [.HEX]   =LED1
Load HEX file'LED1.HEX'8000-8015
File name [.SYM]   =
```

② 在 Z-Vision Remote mini 的打印窗口修改该改的地方；

```
                 80→00                    00→80
ADRS +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F
8000 3E 4F D3 1D 3E 0F D3 1F-3E 00 D3 1E DB 1C E6 40
8010 20 FA 3E 80 D3 1E FF FF-FF FF FF FF FF FF FF FF
```

程序修改处(打印窗口)

③ 运行程序。



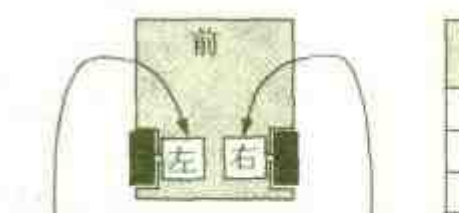


这次最初是 LED 灯亮着、一按下开始开关灯就灭了。其他程序的调试也按与此相同的要领进行即可。

相扑机器人的前进、后退程序



下面来试着控制“太郎”的电机。“太郎”电机的控制数据如右图所示。



	PB1	PB0
前转	0	1
后转	1	0

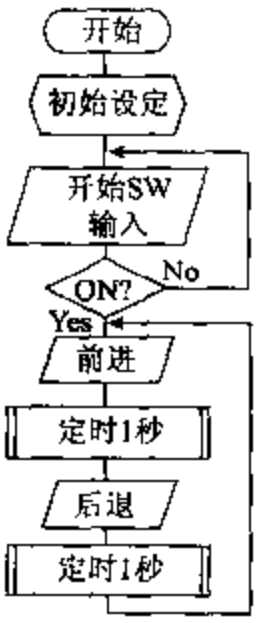
	PB3	PB2
前转	1	0
后转	0	1

太郎	B口数据
前进	09H
后退	06H
右转	05H
左转	0AH
停止	00H

(LED灯灭时)

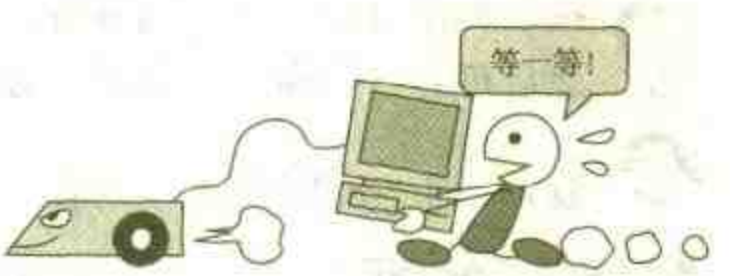


这里来编写前进和后退交替重复的程序。考虑的流程图与程序如下：





```
        ORG 8000H       ;地址设定
        LD A,4FH        ;A 口输入
        OUT(1DH),A
        LD A,0FH        ;B 口输出
        OUT(1FH),A
        LD A,00H        ;B 口清除
        OUT(1EH),A
INKEY:  IN A,(1CH)      ;A 口输入
        AND 40H         ;PA6 屏蔽
        JP NZ,INKEY
START:  LD A,09H        ;前进
        OUT (1EH),A
        LD B,100        ;100 * 10ms 设定
        CALL WAIT       ;调用定时器
        LD A,06H        ;后退
        OUT (1EH),A
        LD B,100        ;100 * 10ms 设定
        CALL WAIT       ;调用定时器
        JR START
 WAIT:  CALL WAIT2      ;定时子程序
        DEC B
        JR NZ,WAIT
        RET
WAIT2:  LD HL,1900H
WAIT3:  DEC L
        JR NZ,WAIT3
        DEC H
        JR NZ,WAIT3
        RET
        END
```



程序中将定时器作为子程序使用，给 B 寄存器输入必要的时间(10ms 为单位)后调用此子程序。



刚才跟老师学了检查程序的步骤。但是按此程序，相扑机器人是移动的，所以 RS-232C 电缆成为障碍。



控制对象在场所内移动时，就会产生此问题。

虽然并不理想，但实际上在 Z-Vision Remote mini 中，即使取掉电缆，程序也会以很高的概率驻留在相扑机器人的 RAM 中。所以，用 Z-Vision Remote mini 运行程序后，可取掉电缆进行实验。

练习题

●试作开始开关按下 5 秒后 LED 灯亮的程序。

提示：将 LED 灯亮程序与定时子程序组合来完成。

10 “太郎”的程序



“太郎”的程序概要

已经学过“太郎”的硬件完成步骤，软件的调试方法等。最后剩下了编写程序的步骤。

按照程序可让“太郎”做各种动作。这里以一个典型程序为例进行说明。之后可各自钻研改良得到更好的程序。

复习一下“太郎”的基本动作

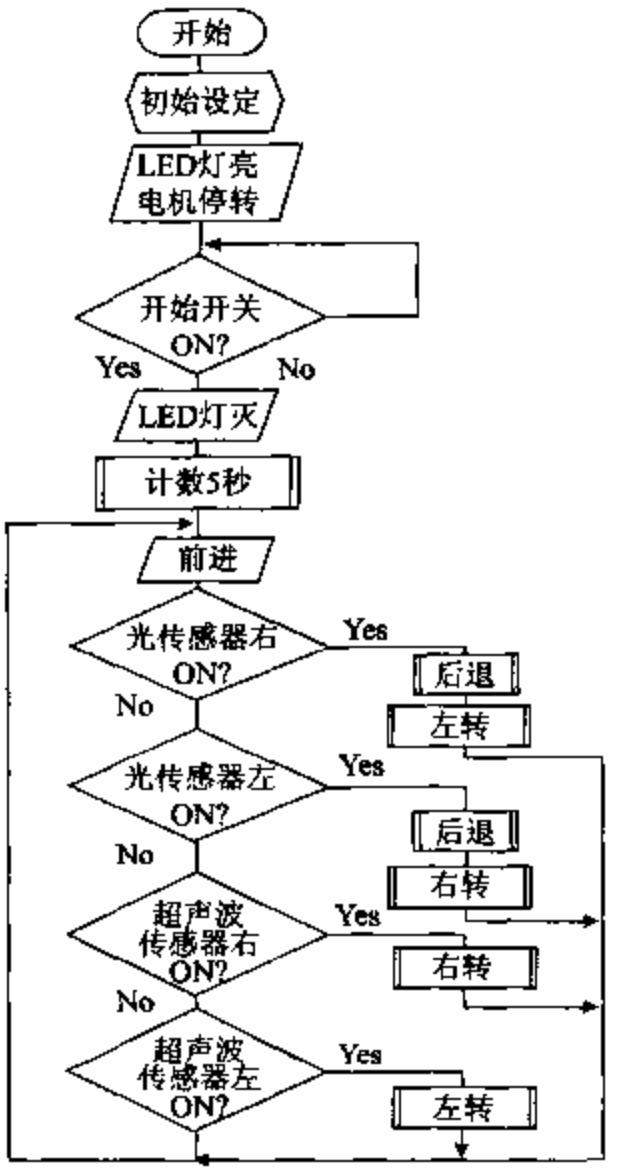
“太郎”的基本动作

① 按下开始开关，5 秒钟后开始前进；
② 在检查光传感器、超声波传感器的同时前进；
③ 光传感器如有反应（检测到摔跤场边框），则稍后退后转向，回到动作②；
④ 超声波传感器如有反应（发现敌人），则旋转 45°，回到动作②。

试着考虑了基本动作流程图。

179 页示出了机器语言的程序例。如果使用插入法等技术，还能更有效地编写程序。这里为使初学者易于理解，未使用较难的技法。

这样，程序稍长一些。



"太郎"的对抗用流程图

初始设定部分



可将程序分成几个部分来考虑。最初即是初始设定部分。初始设定部分还可以分成几个小部分。

(1) 地址设定

程序的开始地址设定为 8000H(RAM 的首地址)。堆栈指针的地址设定在 RAM 终点附近的任意地址处。



程序完成后,将 ORG 8000H 改成 ORG0000H 写入 ROM。

```
;------------------------------
;地址设定
;------------------------------
  ORG 8000H          ;开始地址
  LD SP,0E000H       ;堆栈指针设定
```

(2) PIO 设定

使用 EQU 语句，设定 PIO 的数据地址和口指令。

```
;-------------------------------
;PIO 初始化数据
;-------------------------------
A_DATA      EQU 1CH      A 口数据
A_COMM      EQU 1DH      A 口指令
B_DATA      EQU 1EH      B 口数据
B_COMM      EQU 1FH      B 口指令
IN_BYTE     EQU 4FH      字节输入
OUT_BYTE    EQU 0FH      字节输出
INT_NONE    EQU 07H      无插入
```

指定 A 口为输入用，B 口为输出用。

```
;-------------------------------
;PIO 设定
;-------------------------------
LD A,IN_BYTE          ;A 口字输入
OUT(A_COMM),A
LD A,INT_NONE
OUT(A_COMM),A
LD A,OUT_BYTE         ;B 口字输出
OUT(B_COMM),A
LD A,INT_NONE
OUT(B_COMM),A
```

(3) 数据设定

标明分配给 A 口、B 口各位的器件。

```
;-------------------------------
;A 口分配(输入)
;-------------------------------
PSEN_R      EQU 00H   ;光传感器右
PSEN_L      EQU 01H   ;光传感器左
WSEN_R      EQU 02H   ;超声波传感器右
WSEN_L      EQU 03H   ;超声波传感器左
START_SW    EQU 06H   ;开始开关
;-------------------------------
;B 口分配(输出)
;-------------------------------
;电机左       PB0,PB1
;电机右       PB2,PB3
;LED          PB7
```

设定 LED 和电机控制用的数据。数据会随电机的安装方向以及电压的极性等改变，所以各自要充分检查。

```
;-------------------------------
;电机控制数据(LED灯亮)
;-------------------------------
M_STOP      EQU 80H   ;停止
M_FRONT     EQU 89H   ;前进
M_BACK      EQU 86H   ;后退
M_RIGHT     EQU 85H   ;右转
M_LEFT      EQU 8AH   ;左转
;
M_STOP_OFF EQU 00H   ;停止,LED灯灭
```

如果 LED 灯亮、电机停止，那么认为初始设定部分结束。所以，一打开"太郎"的电源，LED 即呈灯亮状态。

```
;-------------------------------
;LED灯亮,电机停止
;-------------------------------
  LD A,M_STOP
  OUT(B_DATA),A
```

(4) 互相对峙部分

扫描检测开始开关，如果开始开关被按下，则立即调用定时子程序。时间设定用 B 寄存器置为 100，即定时设定为 100 * 10(ms)＝1s。由于定时子程序被调用 5 次，所以其结果是定时 5s。此期间 LED 灯灭，可对动作进行确认。

```
;-------------------------------
;开始等待5s
;-------------------------------
INKEY:IN A,(A_DATA)
      AND 40H
      JR NZ,INKEY
      LD A,M_STOP_OFF
      OUT(B_DATA),A
      LD C,5                    ;5
YET:  LD B,100                  ;5 * 100 * 10(ms)＝5(s)
      CALL WAIT                 ;1(ms)定时子程序调用
      DECC
      JR NZ,YET
```

基本动作部分



此部分为主程序。“太郎”是在扫描检测光传感器与超声波传感器的同时前进的。如果有一个传感器有反应，则调用方向转换子程序。此程序中，各传感器只是检测到1次就有效。但是，如果有小的白色脏东西落入摔跤场内等时，“太郎”就会检测到而不经考虑地转换了方向。也是出于这种考虑，为避免这种麻烦，有效的办法是空一短暂时刻对传感器进行二次检查的方法。



只有传感器两次都反应时才转换方向。程序也要这样编写才好。

```
;-------------------------------
;基本动作
;-------------------------------
ST:     LD A_M_FRONT                    ;前进
        OUT (B_DATA),A
        IN A,(A_DATA)                   ;A口输入
        LD C,A                          ;存入C寄存器
;       ---------------------------------
        BITPSEN_R,C                     ;检测光传感器右
        JR NZ,PLEFT                     ;右有反应左转
        BIT PSEN_L,C                    ;检测光传感器左
        JR NZ,PRIGHT                    ;左有反应右转
;-------------------------------
        BIT WSEN_R,C                    ;检测超声波传感器右
        JR NZ, WRIGHT                   ;右有反应右转
        BIT WSEN_L,C                    ;检测超声波传感器左
        JR NZ,WLEFT                     ;左有反应左转
;       ---------------------------------
        JR ST
;       ---------------------------------
PRIGHT: CALL BACK
        LD B,15
        CALL RIGHT
        JR ST
PLEFT:  CALL BACK
        LD B,15
        CALL LEFT
        JR ST
WRIGHT: LD B,40
        CALL RIGHT
        JR ST
WLEFT:  LD B,40
        CALL LEFT
        JR ST
```

子程序部分



此部分包括后退、右转、左转用的子程序。随置于 B 寄存器中的时间改变旋转角度等。



实际上驱动"太郎",需要决定定时时间。

```
;-------------------------------
;后退子程序
;-------------------------------
BACK:     LD A,M_BACK
          OUT (B_DATA),A
          LD B,60      ;旋转时间设定
          CALL WAIT
          RET
;-------------------------------
;右转子程序
;-------------------------------
RIGHT:    LD A,M_RIGHT
          OUT (B_DATA),A
          CALL WAIT
          RET
;-------------------------------
;左转子程序
;-------------------------------
LEFT:     LD A,M_LEFT
          OUT (B_DATA),A
          CALL WAIT
          RET
```



下边是以 10ms 为单位的定时子程序。置子 HL 寄存器中的数据 1900H 是根据微机板的时钟等计算得来的。

```
;-------------------------------
;定时子程序
;将传值(10ms)放入 B 寄存器后调用
;-------------------------------
WAIT:     CALL WAIT2
          DEC B
          JR NZ,WAIT
          RET
WAIT2:    LD HL,1900H
WAIT3:    DECL
          JR NZ,WAIT3
          DEC H
          JR NZ,WAIT3
          RET
```

```
;-------------------------------------
;相扑机器人"太郎"
;       1996.9.18 堀桂太郎
;-------------------------------------
;地址设定
;-------------------------------------
    ORG 8000H           ;开始地址
    LD SP,0E000H        ;堆栈指针设定
;-------------------------------------
;PIO 初始化数据
;-------------------------------------
A_DATM    EQU 1CH   ;A 口数据
A_COMM    EQU 1DH   ;A 口指令
B_DATA    EQU 1EH   ;B 口数据
B_COMM    EQU 1FH   ;B 口指令
IN_BYTE   EQU 4FH   ;字节输入
OUT_BYTE  EQU 0FH   ;字节输出
INT_NONE  EQU 07H   ;无插入
;-------------------------------------
;PIO 设定
;-------------------------------------
    LD A,IN_BYTE    ;A 口字输入
    OUT (A_COMM),A
    LD A,INT_NONE
    OUT (A_COMM),A
    LD A,OUT_BYTE   ;B 口字输出
    OUT (B_COMM),A
    LD A,INT_NONE
    OUT (B_COMM),A
;-------------------------------------
;A 口分配(输入)
;-------------------------------------
PSEN_R    EQU 00H   ;光传感器右
PSEN_L    EQU 01H   ;光传感器左
WSEN_R    EQU 02H   ;超声波传感器右
WSEN_L    EQU 03H   ;超声波传感器左
START_SW  EQU 06H   ;开始开关
;-------------------------------------
;B 口分配(输出)
;-------------------------------------
;电机左      PB0,PB1
;电机右      PB2,PB3
;LED         PB7
;-------------------------------------
;电机控制数据(LED 灯亮)
;-------------------------------------
M_STOP    EQU 80H   ;停止
M_FRONT   EQU 89H   ;前进
M_BACK    EQU 86H   ;后退
M_RIGHT   EQU 85H   ;右转
M_LEFT    EQU 8AH   ;左转
;
M_STOP_OFF  EQU 00H   ;停止,LED 灯灭
;-------------------------------------
;LED 灯亮,电机停止
;-------------------------------------
    LD A,M_STOP
    OUT (B_DATA),A
;-------------------------------------
;开始等待 5 秒
;-------------------------------------
INKEY:IN A,(A_DATA)
      AND 40H
      JR NZ,INKEY           ;开始开关
                            ;按下否?
      LD A,M_STOP_OFF       ;LED 灯灭,电机停止
      OUT (B_DATA),A
      LD C,5                ;5
YET:  LDB,100               ;5 * 100 * 10(ms)=5(S)
      CALL WAIT             ;1(ms)定时子程序调用
      DECC
      JR NZ,TET
;-------------------------------------
;基本动作
;-------------------------------------
ST:   LD A,M_FRONT          ;前进
      OUT (B_DATA),A
      IN A,(A_DATA)         ;A 口输入
      LD C,A ;存入 C 寄存器
;-------------------------------------
      BITPSEN_R,C     ;检测光传感器右
      JR NZ,PLEFT     ;右有反应左转
      BIT PSEN_L,C    ;检测光传感器左
      JR NZ,PRIGHT    ;左有反应右转
;-------------------------------------
      BIT WSEN_R,C    ;检测超声波传感器右
      JR NZ,WRIGHT;   ;右有反应右转
      BIT WSEN_L,C    ;检测超声波传感器左
      JR NZ,WLEFT     ;左有反应左转
;-------------------------------------
      JR ST
;-------------------------------------
PRIGHT;     CALL BACK
            LD B,15
            CALL RIGHT
            JR ST
PLEFT;      CALL BACK
            LD B,15
            CALL LEFT
            JR ST
WRIGHT;     LD B,40
            CALL RIGHT
            JR ST
WLEFT;      LD B,40
            CALL LEFT
            JR ST
;-------------------------------------
;后退子程序
;-------------------------------------
BACK;       LD A,M_BACK
            OUT (B_DATA),A
            LD B,60         旋转时间设定
            CALL WAIT
            RET
;-------------------------------------
;右转子程序
;-------------------------------------
RIGHT;      LD A,M_RIGHT
            OUT (B_DATA),A
            CALL WAIT
            RET
;-------------------------------------
;左转子程序
;-------------------------------------
LEFT;       LD A,M_LEFT
            OUT (B_DATA),A
            CALL WAIT
            RET
;-------------------------------------
;定时子程序
;将变元(10ms)放入 B 寄存器后调用
;-------------------------------------
WAIT;       CALL WAIT2
            DEC B
            JR NZ,WAIT
            RET
WAIT2;      LD HL,1900H
WAIT4;      DECL
            JR NZ,WAIT3
            DEC H
            JR NZ, WAIT3
            RET
;-------------------------------------
END
"太郎"的对抗用程序例
```



Q 挑战题

① **研究对抗程序**

改良为更有效的动作。

② **研究 5 秒钟立刻启动的动作**

根据对手机器人的形状及性能可选择有效的动作。

③ **研究尽量减小摔跤场地面与机器人间间隙的办法**

防止对手机器人从下面铲、捞。

④ **研究从下面铲、捞对手机器人**

安装兜裆布和吊出臂。

⑤ **研究光传感器的安装位置与数量**

要确切检测出摔跤场的边框并能安全地进行回避。

⑥ **研究超声波传感器的安装位置与数量**

要能确实发现对手机器人，并能进行有效的攻击。

⑦ **研究电机安装位置**

实验是前轮驱动还是后轮驱动等。

⑧ **改为高扭矩电机**

提高对对手机器人的推出力。

⑨ **研究电机速度的提高**

为了以灵敏的动作攻击对手。

⑩ **研究电机速度控制电路**

在摔跤场的中心要高速，靠近摔跤场边框时为防止落入场外要低速。

⑪ **研究轮胎的橡胶及轮胎的大小**

以获得有效的接触摩擦力、移动距离与功率。

⑫ **研究四轮驱动**

增加电机及轮胎使其更有力。

⑬ **低重心化研究**

为了稳定而降低重心。

⑭ **研究各种传感器**

实验研究红外线传感器及接触式传感器等更有效的传感器。

⑮ **动作后增量的研究**

利用抽真空风扇等，使之具有与摔跤场地的吸引力。

⑯ **保护体的研究**

防止让对手机器人的传感器检测出。



练习题

●检测摔跤场边缘白线时的转向及移动时间等怎样确定为好？

解答

由于有电机的位置、旋转速度、轮胎等许多影响因素，因此只能实际上试验机器人的动作，按照其运行错误来确定。

[General Information]
书名=book_11119015
作者=
页数=181
SS号=11119015
出版日期=

封面
书名
版权
前言
目录
目　　录

第 1 章电子控制的基础知识
1 传感器与执行机构
2 光传感器与温度传感器
3 磁传感器与压力传感器
4 电磁螺线管与步进电机
5 伺服电机及其应用
6 接口电路
7　　A - D转换与D - A转换
第 2 章C语言控制入门
1　LED的闪烁控制
2　LED闪烁速度的调整与音阶
3　LED闪烁的缓慢停止
第 3 章计算机控制基础
1 微机控制的种类
2 微机控制的实例
3 微机电路板
4 微机控制的顺序
第 4 章机器人的制作方法
1 机器人相扑比赛大会概况
2 自立型相扑机器人“太郎”
3 微机板的使用方法
4 机械部分的制作方法
5 电子电路的制作方法
6 执行文件的制作方法
7　ROM写入器的使用方法
8 遥控调试程序的使用方法
9 程序调试实践& 1
1 0“太郎”的程序